Operators and Matrices
Volume 9, Number 3 (2015), 697–710 doi:10.7153/oam-09-42

RANDOM WALKS RELATIVE TO MULTIPLE TRANSITION MATRICES

ANTONIJA PRŠLJA

(*Communicated by Jeffrey Hunter*)

Abstract. Given the cost matrix corresponding to transitions between states, the mean of the cost along a random walk of a prescribed length needs to be computed in many applications. We introduce a generalization of the model for multiple transition and cost matrices and propose Monte Carlo techniques to solve it. Experiments on artificial data and a small example of simulated bike sharing system are conducted to evaluate the performance of the presented approaches in comparison with the ones based on computing powers of matrices.

1. Introduction

Let $S = \{s_1, s_2, \dots, s_n\}$ be a finite set of states, and let P be the $n \times n$ transition matrix with (i, j)-entry equal to the transition probability $p_{ij} \in [0,1]$ from state s_i to s_j. In discrete time $0, 1, \dots, k$, starting at a specified state s_{j_0}, a particle travels between states such that, at each time step, the next state to be visited is chosen according to the transition matrix P: if the current state is s_i, then the particle moves to the state s_j with the probability p_{ij}. The (random) sequence $W^P = s_{j_0}s_{j_1}\cdots s_{j_k}$ of states selected this way is a *random walk* of length k governed by P. In the literature it is also referred to as a discrete-time Markov chain [21]. Its probability p_{W^P} is equal to $p_{j_0 j_1} p_{j_1 j_2} \cdots p_{j_{k-1} j_k}$. Further, let C be the $n \times n$ cost matrix with (i, j)-entry equal to the cost $c_{ij} \in \mathbb{R}$ of the transition from state s_i to state s_j. Then the cost along $W^P = s_{j_0}s_{j_1}\cdots s_{j_{k-1}}s_{j_k}$ is $c_{W^P} = c_{j_0 j_1} + c_{j_1 j_2} + \cdots + c_{j_{k-1} j_k}$. Let the random variable X denote the cost along a random walk W^P of length k starting at s_{j_0}, and let $S^k_{j_0}$ be the set of all walks of length k starting at s_{j_0}. The mean

$$E(X) = \sum_{W^P \in S^k_{j_0}} c_{W^P}\, p_{W^P}$$

of a random variable X needs to be computed in many applications.

For exact computation of the mean there is the problem that in general the cardinality of the set of all random walks of length k starting at s_{j_0} grows exponentially

Mathematics subject classification (2010): 60J22.
Keywords and phrases: Multiple transition matrices, cost matrix, Monte Carlo method, random walk, discrete Markov chain.

This paper is presented on 23rd International Workshop on Matrices and Statistics, and 7th Linear Algebra Workshop, Ljubljana, Slovenia, June 8-12, 2014.

Operation part financed by the European Union, European Social Fund.

This paper was written while working for company Arctur d.o.o., Nova Gorica, Slovenia.

with k. What is more, in practice the cardinality of S can be large as well. Thus, with increasing k and n, explicit computation of the mean becomes impractical in terms of both time and space. It is then reasonable to provide a way to approximate the mean. Describing efficient methods for estimating the mean of X is the main objective of this paper.

This problem has been briefly discussed in [14]. Their approach involves taking an appropriate integral form, discretizing it within some specified error, and then computing each term of discretization, again within a specified error, using diffusion wavelets method for computing powers of matrices that is proposed in [7]. It is important to stress that diffusion wavelets method behaves satisfactorily when a matrix has low numerical rank and a power is of the form 2^p for some positive integer p.

In this paper, a different approach is proposed that can be used for an arbitrary transition matrix as well as for an arbitrary k. The key idea is to exploit Monte Carlo techniques that allow us to get approximate value by using random numbers and simulation [21]. More about Monte Carlo approximation and the computation of the expected cost can also be found in [21]. These techniques are very commonly used in many different fields, such as physics, physical chemistry, computational biology, finance [13] and Bayesian statistical inference [12]. For more information on Monte Carlo methods, integration and Markov chain Monte Carlo methods we refer the reader to [17, 4, 6, 20].

The intention of this paper is twofold. Firstly, to introduce a generalization of the above model for multiple transition and cost matrices. And secondly, to provide different possible approaches for solving the basic as well as the generalized problem. As already mentioned, the key step in all our approaches is to use Monte Carlo techniques.

The rest of the paper is structured as follows. In Section 2 we present a motivation application. In Section 3 we give a formal description of generalized model. In Section 4 we devise approaches for solving the basic problem as well as the general case. A detailed description of the approach from [14] is presented in Section 5. Section 6 presents performances of our approaches in comparison with that from [14] over some synthetic data, along with the evaluation results. Possible future work is discussed in Section 7.

2. Motivation application

In various big cities, such as Paris, Ljubljana, Venice, bike sharing systems have become very popular way of transport. Each city has certain number of bike stations that are distributed around the city and each station contains a fixed number of bikes. A registered user can pick up a bike at any station and return it at any other station where a stand is available for a bike to be locked. Rides of different bicycles are repeated a number of times daily by different users. Availability of bikes or lockers at any station is different at a different period of a day.

When a bike user comes to desirable station to return (or rent) a bike, there should be enough free lockers (or bikes) to return (or rent) the bicycle. This problem in bike sharing systems is dealt with a redistribution of bikes using trucks to rebalance bikes between stations that are emptying and those that are filling up and with this action allowing the users to rent and return the bicycles to their desired station (for more see [23, 5, 8] and references therein). The problem of demand for bicycles at each station is similar to the problem of available beds in hospitals, where the patients are requested to go elsewhere because of hospital overcrowding [3, 2]. Also the problem of deciding how to assign movies to multiple disks (servers) known as video-on-demand system is discussed in [15, 16] and is quite similar to bikes being transported to different stations.

In this paper, we consider the following problem in bike sharing system. Suppose, for every bicycle, that we know the exact time and station at which it was picked up and returned back. We observe their rides and after sufficiently long period of time we calculate their frequencies of their movements from each station to any other station. We can than construct the transition matrix from gathered data. The use of bikes is usually distributed differently in the morning, afternoon, or in the evening. Treating rides of all bicycles in three time periods per day separately, we can construct three different transition matrices. The corresponding cost matrices may be, for example, the distance between stations on marked bicycle routes. We are interested in average mileage of a bike after a given number of rides with transitions determined by three transition matrices.

3. Generalization of the basic model

Now we give a formal description of the generalized problem. Suppose we have transition matrices P, P' along with cost matrices C, C', and that in discrete time $0, 1, \ldots, (l+l')k$, starting at s_{j_0}, a particle makes first l steps with transition probabilities in P, then next l' steps with transition probabilities in P', and then again l steps with transition probabilities in P, and so on. This constitutes a random walk

$$W^{P,P'} = \prod_{j=1}^{k} W_{2j-1}^{P} W_{2j}^{P'} \tag{1}$$

of length $(l+l')k$ governed by P and P'. Then its probability is equal to $p_{W^{P,P'}} = \prod_{j=1}^{k} p_{W_{2j-1}^{P}} p_{W_{2j}^{P'}}$, while its cost is $c_{W^{P,P'}} = \sum_{j=1}^{k} c_{W_{2j-1}^{P}} + c_{W_{2j}^{P'}}$. Let the random variable Y denote the cost along random walk $W^{P,P'}$ of the form (1) with initial state being s_{j_0}. We are interested in efficiently estimating the mean

$$E(Y) = \sum_{W^{P,P'} \in S_{j_0}^{(l+l')k}} c_{W^{P,P'}} \, p_{W^{P,P'}}. \tag{2}$$

We generalize this idea in obvious way using multiple random transition matrices.

4. Monte Carlo approach

We shall describe how to use Monte Carlo approach in order to compute the estimate of the mean of a random variable. We shall first discuss the case in which we have only one transition matrix, and deal with the general case later.

4.1. Basic algorithm for one transition matrix

Let P be a given transition matrix, k a positive integer and s_{j_0} an initial state. As is usually the case in Monte Carlo approach, we generate N random walks $W_1^P,\dots,W_N^P$ of length k starting at s_{j_0}, compute their costs $c_{W_1^P},\dots,c_{W_N^P}$, and get the estimate

$$E(X) \approx \frac{1}{N}\sum_{i=1}^{N} c_{W_i^P}. \tag{3}$$

In order to generate a random walk W^P of length k, we describe how to construct a new state. First, we calculate cumulative matrix Q according to P: every row i of matrix Q is a cumulative distribution function of every row i of matrix P. Suppose s_i is the current state. We generate a random number $r \in [0,1]$. Let a_i be the index such that $Q[i,a_i] > r$ and $Q[i,a_i-1] \leqslant r$. In other words, a_i is such index that $Q[i,a_i] > r$ holds for the first time. The index of the new state is set to a_i. Starting at s_{j_0} and repeating the above construction k times we obtain a random walk at s_{j_0} of length k. The cost along the constructed walk is computed simultaneously with the construction itself. Formal code for simulation of a random walk and its cost along is given in the COSTWALK. Using the COSTWALK we generate N random walks $W_1^P,\dots,W_N^P$ of length k starting at state s_{j_0}, compute their costs and get the estimate $\hat{\mu}$ for $\mu = E(X)$ as in (3).

```
COSTWALK
Input:  $P \in \mathbb{R}^{n\times n}$ {transition matrix},
        $C \in \mathbb{R}^{n\times n}$ {cost matrix corresponding to transitions between states},
        $j_0 \in \{1,\dots,n\}$ {initial state $s_{j_0}$},
        $k \in \mathbb{Z}^+$ {length of random walk}
Output: $c \in \mathbb{R}$ {Cost along generated random walk $W^P$ of length $k$ starting at state $s_{j_0}$.}
  Compute cumulative matrix $Q \in \mathbb{R}^{n\times n}$ according to $P$.
  Set $i = j_0$ and $c = 0$.
  for $j = 1$ to $k$ do
    Generate a random number $r \in [0,1]$.
    Let $a_i$ be the index such that $Q[i,a_i] > r$ for the first time.
    Set $c = c + C(i,a_i)$.
    Set $i = a_i$.
  end for
  Return $c$.
```

For considering the accuracy of the COSTWALK, note that as the Monte Carlo method is a stochastic method, different runs of the Monte Carlo method typically lead to different results. Thus, we have to deal with stochastic error. Let σ denote the standard deviation of X. In practice σ is almost never known a priori. For this purpose, we estimate it with $S = \sqrt{1/(N-1)\sum_{i=1}^{N}(c_{W_i^P} - \hat{\mu})^2}$ and then from the central limit theorem we obtain an approximate $(1-\alpha)$-confidence interval for μ

$$\left[\hat{\mu}-t^{*}\frac{S}{\sqrt{N}},\, \hat{\mu}+t^{*}\frac{S}{\sqrt{N}}\right]. \tag{4}$$

The parameter t^{*} is known and is determined by $P(-t^{*} \leqslant T \leqslant t^{*}) = 1-\alpha$, where random variable T has a Student's t-distribution with $N-1$ degrees of freedom.

Now we need to find the appropriate number of simulations N in terms of the desired accuracy and the confidence interval on the accuracy. Assume that one desires a final approximate confidence interval with half-width $\varepsilon|\mu|$, that is, with relative error being ε. It follows then from (4), that $\varepsilon|\mu| = (t^{*}S)/\sqrt{N}$. Finally, since $\mu \approx \hat{\mu}$, we get an approximation for N

$$N \approx \left(\frac{t^{*}S}{\varepsilon\hat{\mu}}\right)^{2}. \tag{5}$$

In practice the number N is typically selected by running a small number of trials runs N_0. The value t^{*} is calculated corresponding to N_0 and given α. First, we use the COSTWALK to generate random walks $W_1^P, W_2^P, \ldots, W_{N_0}^P$ and to compute their costs $c_{W_1^P}, c_{W_2^P}, \ldots, c_{W_{N_0}^P}$. Then we calculate estimate $\hat{\mu}_0$ and sample variance S_0^2 to get approximation (5) for N. Finally, for computed N we execute the COSTWALK to generate N new random walks. For simplicity let BALG denote the basic algorithm for one transition matrix in other words Monte Carlo method using COSTWALK to generate a random walk.

REMARK 1. Consider the problem of one transition matrix. Then the random variable X can be written as a sum over transitions as follows. Let $T_{ij}(W^P)$ be the random variable denoting the number of transitions from the state s_i to the state s_j in a specific random walk W^P. Then $X = \sum_{i=1}^{n}\sum_{j\neq i}^{n} c_{ij}T_{ij}(W^P)$. To calculate $E(X)$ we now need to calculate $E(T_{ij}(W^P))$. Taking into account the condition on the final state s_{j_k} of W^P we can write $E(T_{ij}(W^P))$ as the sum of conditional means $E(T_{ij}(W^P)\,|\,j_k)$, where j_k ranges over all indices $1,2,\ldots,n$. Of course, explicit computations of such conditional means are rather hopeless for large n and k, and thus we would like to at least find approximations. For this purpose, we can apply different algorithms for endpoint conditional simulation from Markov chains [11, 18, 19].

REMARK 2. Consider the problem of one transition matrix and suppose we estimate the mean $E(X)$ using Monte Carlo method. Instead of using Monte Carlo method one would think of using the variance reduction method called importance sampling. This method tries to produce an estimator for $E(X)$ with smaller variance than the one provided with Monte Carlo method. But importance sampling is quite inefficient in high-dimensional spaces because the variance of the likelihood ratio blows up [1].

4.2. Use of basic algorithm for multiple transition matrices

We use the same idea to solve the generalized problem. To be precise, given transition matrices P, P' and their cost matrices C, C', we generate N random walks

$W_1^{P,P'},\dots,W_N^{P,P'}$ of length $(l+l')k$ starting at initial state s_{j_0}, calculate their costs, and estimate (2) as:

$$E(Y) \approx \sum_{i=1}^{N} c_{W_i^{P,P'}}. \tag{6}$$

We give three different approaches how to generate random walks of the form (1) and compute their costs. In some way or another, all these approaches exploit the COSTWALK.

First approach (APP1). We first calculate cumulative matrices Q, Q' according to P, P'. Then from the COSTWALK we use construction to select the new state from a current state. First l states are selected according to cumulative matrix Q and next l' states according to Q'. Repeating this k times we obtain a random walk $W^{P,P'}$ starting at s_{j_0} of length $(l+l')k$ and its cost $c_{W^{P,P'}}$. Then we generate N random walks $W_1^{P,P'},W_2^{P,P'},\cdots,W_N^{P,P'}$ of length $(l+l')k$ starting at state s_{j_0}, compute their costs and get the estimate $\hat{\mu}$ in (6).

Second approach (APP2). First, for transition matrices P, P' we calculate their powers P^l, $(P')^{l'}$ and cumulative matrices Q, Q' according to P^l, $(P')^{l'}$. For new transition matrices P^l, $(P')^{l'}$ we calculate their cost matrices C_l, $C_{l'}$ as follows. For every state s_i, $i=1,2,\dots,n$, we find all possible walks starting at s_i of length l using modified breadth-first search [9]. Let s_i be an arbitrary initial state. With modified breadth-first search we find all walks starting at s_i of length l and for every walk we store finite state. Let $W_1^P,W_2^P,\cdots,W_u^P$ be all walks of length l starting at s_i and having finite state j. We can define cost c_{ij}^l from state s_i to state s_j for (i,j)-entry of cost matrix C_l according to transition matrix P as

$$c_{ij}^l = p_{W_1^P}c_{W_1^P} + p_{W_2^P}c_{W_2^P} + \cdots + p_{W_u^P}c_{W_u^P}. \tag{7}$$

Using formula (7) we calculate every entry of cost matrix C_l according to transition matrix P. The same way we calculate cost matrix $C_{l'}$ according to P'. Now we have cost matrices C_l, $C_{l'}$ according to $F=P^l$ and $F'=P^{l'}$. Again using construction from the COSTWALK we select first state according to cumulative matrix Q and second state according to cumulative matrix Q'. Repeating this for k times we obtain random walk $W^{F,F'}$ of length $2k$ and its cost $c_{W^{F,F'}}$. Then we generate N random walks, their cost and we get the estimate $\hat{\mu}$ in (6).

Third approach (APP3). As in second approach we first calculate $F=P^l$, $F'=(P')^{l'}$ and their corresponding cost matrices C_l, $C_{l'}$. We calculate new transition matrix $Z = F\cdot F'$ and its corresponding cost matrix C_Z. As in second approach, for every state we find all possible walks of length 2 using modified breadth-first search. We use formula (7) to calculate every element of cost matrix C_Z. Now using the COSTWALK we generate N random walks $W_1^Z,W_2^Z,\cdots,W_N^Z$ of length k and their cost $c_{W_1^Z},c_{W_2^Z},\cdots,c_{W_N^Z}$ we get the estimate in (6).

All three approaches can be generalized for three or more transition matrices. In the third approach we can also apply approach using diffusion wavelets method presented in [14], where the diffusion wavelets method by [7] is used for calculating dyadic powers of transition matrix Z.

5. Approach using diffusion wavelets method (ADW)

In this section we discuss another approach which was proposed in [14] and is based on diffusion wavelets method for computing powers of matrices treated in [7]. Later we compare method ADW with our methods introduced in Section 4.

Let $f:\mathbb{R}\to\mathbb{R}$ be a function and let X be a random variable as defined above. For implementation purpose, we here describe, in some details, the approach for estimating

$$E(f(X)) = \sum_{j_1,j_2,\dots,j_k\in S} f(c_{j_0j_1}+\dots+c_{j_{k-1}j_k})\, p_{j_0j_1}\cdots p_{j_{k-1}j_k}, \tag{8}$$

that is proposed in [14].

First, the idea is to rewrite the sum into an integral form. Let $P_C(t)$ be $n\times n$ matrix function of real t with the elements of the form $\exp(\mathbb{i}c_{ij}t)p_{ij}$. Denote by $\mathbf{1}$ a $n\times 1$ column vector of ones and by $e_{j_0}^T$ a $1\times n$ row vector with the j_0-th component equal to 1, and all the others 0. Note that (u,v)-th entry of $n\times n$ matrix $[P_C(t)]^k$ is of the form $\sum_{j_1,j_2,\dots,j_{k-1}}\exp(\mathbb{i}(c_{uj_1}+c_{j_1j_2}+\dots+c_{j_{k-1}v})t)\; p_{uj_1}p_{j_1j_2}\cdots p_{j_{k-1}v}$. By computation we obtain that

$$e_{j_0}^T[P_C(t)]^k\mathbf{1} = \sum_{j_1,j_2,\dots,j_{k-1},j_k\in S} \exp(\mathbb{i}(c_{j_0j_1}+\dots+c_{j_{k-1}j_k})t)p_{j_0j_1}\cdots p_{j_{k-1}j_k}. \tag{9}$$

Using non-unitary inverse Fourier transform $f(x)=\frac{1}{2\pi}\int_{-\infty}^{\infty}\hat{f}(t)\exp(\mathbb{i}xt)dt$ together with (9) we see that

$$E(f(X)) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\hat{f}(t)e_{j_0}^T[P_C(t)]^k\mathbf{1}dt. \tag{10}$$

Having written $E(f(X))$ in an integral form the authors in [14] further suggest to discretize it within some specified error and then compute each term of discretization, again within a specified error, using diffusion wavelets approach given in [7].

Taking $f(x)=x$ we get our initial problem. In this case the Fourier transform is $\hat{f}(t)=2\pi\mathbb{i}\delta'(t)$, where δ' is a first derivative of Dirac delta function. Since Dirac delta function is zero everywhere except at zero, the discretization is trivial. Using the following property $\int_{-\infty}^{\infty}\delta'(t)g(t)dt=-g'(0)$, we have that

$$E(X) = e_{j_0}^T\left[\sum_{i=1}^{k}P_C(0)^{i-1}(DP_C(t))\big|_{t=0}P_C(0)^{k-i}\right]\mathbf{1}, \tag{11}$$

where D denotes the operator $\frac{1}{\mathbb{i}}\frac{d}{dt}$, with the elements of matrix $DP_C(t)$ being equal to $\exp(\mathbb{i}c_{ij}t)c_{ij}p_{ij}$.

As suggested in [14] let $k=2^p$ for some $p>0$. Then the sum (11) can be reduced to

$$E(X) = e_{j_0}^T\sum_{i=1}^{k/2}\left(P_C(0)^{i-1}DP_C(t)|_{t=0}P_C(0)^{k-i}+P_C(0)^{k-i}DP_C(t)|_{t=0}P_C(0)^{i-1}\right)\mathbf{1}. \tag{12}$$

Powers $P_C(0)^j$ of matrix $P_C(0)$ in the upper sum are calculated as follows. First, the power j is written as the sum of powers of 2: $j = 2^{p_1} + 2^{p_2} + \cdots + 2^{p_r}$. Then each dyadic power of $P_C(0)$ is computed as in [7]. Finally, we compute $P_C(0)^j$ as $P_C(0)^j = P_C(0)^{2^{p_1}} P_C(0)^{2^{p_2}} \cdots P_C(0)^{2^{p_r}}$. Note that for computing the largest power $P_C(0)^{2^p-1}$ we need to compute all dyadic powers $P_C(0)^{2^i}$ for each $i = 1,\dots,p-1$. In order to save time we therefore first compute all dyadic powers and then store them so they can be reused again when computing other powers.

6. Experiments

In order to demonstrate the capabilities of all proposed approaches BALG, APP3, APP2, and APP1 we compared their performances with approach ADW. We have implemented all approaches in MATLAB. In our implementation of ADW we used already implemented MATLAB code for computing dyadic power of a matrix [7].

6.1. Data simulation

For experimental purpose, we generated a database, consisting of transition and cost matrices. We first describe how we generated a database for artificial examples and for example describing bike sharing system in Ljubljana.

Artificial database.

Random transition matrix. We generate $n \times n$ random transition matrix P such that each entry is nonzero with probability $\frac{1}{\sqrt{n}}$, and for each i, $\sum_j p_{ij} = 1$. Further, we require that every element of matrix P^n is nonzero. In our approaches we can use transition matrix generated as described above. However, the ADW approach requires matrices with low numerical rank, because it uses diffusion wavelets method. For this purpose, we do additional test in order to ensure that a simulated transition matrix has low numerical rank with some specified tolerance.

Cost matrix. For every nonzero (i,j)-entry of P we describe how to generate the (i,j)-entry of the corresponding cost matrix C. Of course, if the (i,j)-entry of P is equal to zero, then the (i,j)-entry of C is zero. We define a random variable $Y = c \cdot |\log(X)| - d$, where $c,d > 0$, $X \sim U(0,1)$ is the continuous uniform random variable. Then $c|\log(X)| \sim \exp(\frac{1}{c})$. For x generated from continuous uniform distribution we generate (i,j)-entry of C as the value generated from exponentially distributions with parameter $\frac{1}{c}$ and shift that value for a constant $-d$, so some of the entries have negative value.

As explained above, we prepared the database including two basic cases with single transition and cost matrix, namely one of dimension 411×411 and another of dimension 1000×1000, and two general cases with double transition and cost matrices, namely one of dimension 235×235 with $l = 2$, $l' = 3$ and another of dimension 500×500 with $l = 2$, $l' = 1$. As for simulating cost matrices, in all four artificial cases the following values of parameters were selected: $c = 100$ and $d = 10$.

Bike database.

We present an example of a bike sharing system in Ljubljana, Slovenia, where currently there are 36 bike stations.

Transition matrix. The bicycles demand is distributed so that the stations in the city center are mostly arrival stations during the morning (and early afternoon) and mainly departure stations in the late afternoon (and the rest of the day). We assume that most people tend to travel for short distances, so the longer trips are less likely to occur. In light of these assumptions we simulate two transition matrices describing demand of bikes in the morning and the late afternoon. In the morning the probability from stations in suburban areas to stations in the city center are higher then the probability for a bike to travel from the center to suburban stations. The reverse happens in the afternoon. More precisely, we first generate two random matrices Q_1 and Q_2. Each entry q_{ij} of both matrices Q_1 and Q_2 is drawn from the uniform distribution on the interval $[a,b]$, where the interval $[a,b]$ is chosen according to locations of the stations. For this purpose, we cluster all stations into five groups: the center group (in Figure 1 the group of stations marked with a circle) and four suburban groups (in Figure 1 groups of stations marked with a cross, a triangle, a square, and a star, respectively). The intervals are then determined as follows:

- if both stations are in the center group, then $a = 0.6$ and $b = 0.9$,
- if one station is in one suburban group, while the other is in the another suburban group, then $a = 0$ and $b = 0.01$,
- if both stations are in the same suburban group, then $a = 0$ and $b = 0.2$.

Also, the probability for a bike to return back to the same station is nonzero ($a = 0.01$ and $b = 0.35$). In the morning, transitions between stations from the center (suburban) to the suburban (center) stations are drawn on the interval $[0, 0.3]$ ($[0.7, 1]$). The reverse happens in the afternoon. With Q_1 and Q_2 determined, we then normalize each row of Q_1 (Q_2) and obtain the transition matrix P_1 (P_2) for the morning (afternoon) movements of bikes.

Notice that for this real application the transition matrices have all entries nonzero and we can not guarantee that the matrices have low numerical rank as ADW method requires.

Cost matrix. In this case the cost matrix is a matrix of distances between stations on marked bicycle routes. To determine distances we used google map to find the shortest paths between all stations. Note that these paths are not air distances but are geographical routes between stations. Since the probability for a bike to return back to the same station is nonzero, we generate a random distance on the interval $[1.1, 2.2]$ for the values c_{ii}.

6.2. Comparison and evaluation

All approaches have been compared with respect to execution time. Experimental results are gathered in Tables 1–3. The first column shows the length k of generated

Table 1: *Performance comparison of BALG and ADW methods for a transition matrix and its corresponding cost matrix both of dimensions* 411×411*, where* $N_0 = 10^4$.

k	Alg.	$\hat{\mu}$	t(s)	N
2^5	BALG	2867.0076	7	252622
	ADW	2869.9788	204	-
2^6	BALG	5761.6416	7	124962
	ADW	5762.4474	252	-
2^7	BALG	11501.5905	7	64222
	ADW	11504.7264	301	-
2^8	BALG	23057.6052	7	31737
	ADW	23048.1894	366	-
2^9	BALG	46138.7667	7	15791
	ADW	46114.1666	420	-
2^{10}	BALG	92259.0426	7	7972
	ADW	92243.1721	515	-

Table 2: *Performance comparison of BALG and ADW methods for a transition matrix and its corresponding cost matrix both of dimensions* 1000×1000*, where* $N_0 = 10^4$.

k	Alg.	$\hat{\mu}$	t(s)	N
2^5	BALG	2907.3761	903	25980000
	ADW	2908.1305	1806	-
2^6	BALG	5796.9135	1025	12892788
	ADW	5797.5404	2167	-
2^7	BALG	11640.7266	1111	6604388
	ADW	11639.3477	2715	-
2^8	BALG	23255.1055	1085	3211753
	ADW	23255.3864	3215	-
2^9	BALG	46532.6303	1115	1627952
	ADW	46531.0292	3962	-
2^{10}	BALG	92963.1209	1090	806173
	ADW	92967.9280	5336	-

random walks, while the second one describes possible methods: BALG, APP1, APP2, APP3, ADW. The third column represents results $\hat{\mu}$ returned by the methods. Execution times given in seconds for each method are displayed in the fourth column. The last column presents the number of trials N for BALG, APP1, APP2 and APP3. The initial number of trials N_0 is specified for each example separately. According to our work, we use the following values of parameters: $\alpha = 0.01$ (in all cases), $\varepsilon = 10^{-3}$ (in the first artificial case) and $\varepsilon = 10^{-4}$ (in all the other artificial cases). We have another two columns in the case of a bike sharing system, one presenting the initial number of trials N_0 and the other presenting the relative error ε for methods BALG, APP1, APP2, APP3. The precision used in diffusion wavelet method is the number $2.2 \cdot 10^{-16}$ integrated in MATLAB.

As can be seen from Tables 1–2, it is clear that BALG outperformed the ADW method. For a result $\hat{\mu}$ in Table 1 calculated with BALG, there is a 1% probability that the value $\hat{\mu}$ differs from the true value in a third digit ($\varepsilon = 10^{-3}$). In Table 2 the value $\hat{\mu}$ differs from the true value in the fourth digit ($\varepsilon = 10^{-4}$) with probability 1%.

Considering approximations calculated with two methods ADW and BALG, it can be seen that the results of both are very similar but the ADW method has longer execution time. With k increasing execution times of ADW also increase, while the execution times of BALG are rather constant for specified relative error ε and fixed uncertainty α.

The value $\hat{\mu}$ in Tables 3 differs from the true value in the fourth digit ($\varepsilon = 10^{-4}$) with probability 1% for APP1, APP2, APP3 methods. Between the results calculated with our approaches and ADW there are relatively small deviation.

Considering execution times, it is clear that APP3 outperformed all other approaches APP1, APP2, and ADW. The APP3 method is faster then APP2 and APP1, because it generates N random walks of length k, while APP2 and APP1 generate N random walks of length $2k$ and $(l+l')k$, respectively. For small k and n ADW approach is better then APP1 and APP2, however, they become faster then ADW with k and n increasing.

For the last artificial example we tried to find the error which gives the break-even point so that all methods are comparable in time.

Table 3: *Performance comparison of APP1, APP2, APP3 and ADW methods for double transition matrices and its corresponding cost matrices of dimensions* 500×500 *with* $l = 2$, $l' = 1$, *where* $N_0 = 10^3$.

k	Alg.	$\hat{\mu}$	t(s)	N
2^{10}	APP1	275223.5792	864	286466
	APP2	275223.1383	416	226185
	APP3	275217.7672	28	38750
	ADW	275219.5776	603	-
2^{11}	APP1	550461.3854	812	132737
	APP2	550477.7314	421	111098
	APP3	550483.9229	30	19300
	ADW	550477.8523	936	-
2^{12}	APP1	1100930.7115	850	69922
	APP2	1100930.7105	444	54899
	APP3	1100917.4028	30	9650
	ADW	1100967.0968	1397	-

Table 4: *Performance comparison of APP1, APP2, APP3 and ADW methods for double transition matrices and its corresponding cost matrices of dimensions* 235×235 *with* $l = 2,\ l' = 3$.

k	Alg.	$\hat{\mu}$	t(s)	N	N_0	ε
2^{10}	APP1	461648.4380	61	18409	10^4	$2.93 \cdot 10^{-4}$
	APP2	461647.1741	62	55591	10^4	$1.13 \cdot 10^{-4}$
	APP3	461692.2653	61	103842	10^5	$2.55 \cdot 10^{-5}$
	ADW	461695.8276	63	-	-	-
2^{11}	APP1	923446.5778	93	14765	10^4	$2.33 \cdot 10^{-4}$
	APP2	923465.0158	99	46237	$2 \cdot 10^4$	$8.74 \cdot 10^{-5}$
	APP3	923424.5938	94	78929	$3 \cdot 10^5$	$2.06 \cdot 10^{-5}$
	ADW	923421.5854	96	-	-	-
2^{12}	APP1	1846831.9009	162	11630	10^4	$1.85 \cdot 10^{-4}$
	APP2	1846800.4467	169	29525	10^4	$7.75 \cdot 10^{-5}$
	APP3	1846833.2154	167	60360	$3 \cdot 10^4$	$1.66 \cdot 10^{-5}$
	ADW	1846830.5176	168	-	-	-

Table 5: *Performance comparison of APP1, APP2, APP3 and ADW methods for double transition matrices and its corresponding cost matrices of dimensions* 36×36 *with* $l = 7,\ l' = 5$.

k	Alg.	$\hat{\mu}$	t(s)	N	N_0	ε
2^8	APP1	4856.7788	0.415721	282	10^2	$2.2 \cdot 10^{-3}$
	APP2	4854.4233	0.463033	2697	10^3	$3.4 \cdot 10^{-4}$
	APP3	4855.6444	0.432111	4810	10^3	$2.4 \cdot 10^{-4}$
	ADW	4855.0656	0.588918	-	-	-
2^{10}	APP1	19416.1202	0.457121	76	10	$2.5 \cdot 10^{-3}$
	APP2	19421.8255	0.451192	652	10^2	$3.5 \cdot 10^{-4}$
	APP3	19421.3344	0.412310	1135	10^3	$2.5 \cdot 10^{-4}$
	ADW	19422.7111	0.983146	-	-	-
2^{12}	APP1	77659.5380	1.391270	59	10^2	$1.5 \cdot 10^{-3}$
	APP2	77696.4150	1.015229	372	$2 \cdot 10^2$	$2.3 \cdot 10^{-4}$
	APP3	77696.7678	1.532646	1096	$4 \cdot 10^2$	$1.3 \cdot 10^{-4}$
	ADW	77696.7809	2.333481	-	-	-

Result for average mileage of a bike starting at some specified station after a given number of rides are presented in Table 5. Theoretically we have already noted that the number of trials for APP1, APP2, APP3 methods using Monte Carlo techniques grow as the error bound decreases. However, in real time rather than finding very accurate estimates for an average mileage (or mean cost) it is more important to get a quick approximation. For instance, if a bike should be taken to maintenance or service for repair after making some specified average mileage it is not important, if we make a one percentage error. Of course, there are fields where one might need more accurate estimate and where ADW method could be more useful than Monte Carlo methods.

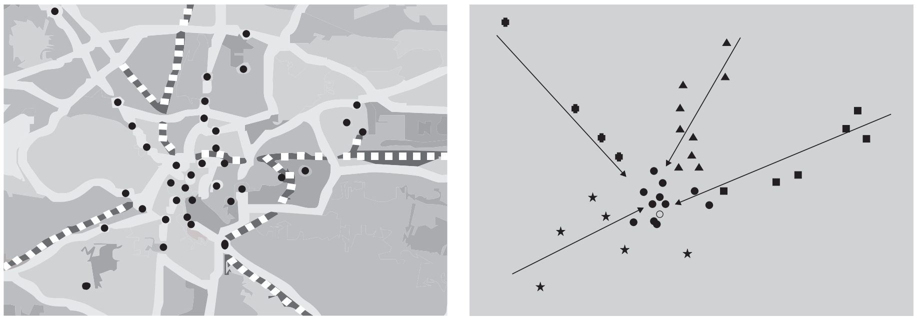
Figure 1: *Left figure presents real positions of bike stations and main rows in Ljubljana, while right figure presents division into five subareas.*

In bigger cities like Taiyuan (1000 stations), Montreal (411 stations), Changwon (235 stations), Barcelona (424 stations) which have large bike sharing networks the methods APP1, APP2 and APP3 are more useful than ADW regarding computational time needed for calculating the average mileage. That can be seen from our artificial examples, where APP3 outperformed ADW in all cases.

7. Further work

In this section we shall briefly discuss our future work. Although Monte Carlo methods are conceptually simple, versatile, flexible, and easy to use, their slow convergence can be a deficiency in computations involving high accuracy. By the central limit theorem, the rate of convergence is roughly of order $N^{-1/2}$ in the sample size N undertaken. In order to improve the convergence, a possible strategy is to replace random numbers by quasirandom numbers. This leads to Quasi-Monte Carlo methods [22, 10] that can achieve convergence of order N^{-1} in certain cases.

In generalized model we used a constant number of steps l and l' relative to transition matrices P and P'. Applying this model to bike sharing system it would be more interesting to have a variable number of rentals for different time periods within a day. Implementing this for APP1 method it is not so difficult, if we can choose l, l' successively from some given prior distribution. Modifying methods APP2 and APP3 in this direction will slightly increase computation time and may increase memory space, while the method ADW most likely can not be modified for l and l' constantly changing. Our further work will concentrate in that direction.

Acknowledgements. The author would like to thank Damjana Kokol Bukovšek, Matjaž Omladič and Gregor Šega for the profound discussions and guidance. Operation part financed by the European Union, European Social Fund.

REFERENCES

[1] S. ASMUSSEN AND P. W. GLYNN, *Stochastic Simulation: Algorithms and Analysis*, Springer, New York, (2007).

[2] K. AU, G. BYRNES, C. BAIN, M. FACKRELL, C. BRAND, D. CAMPBELL AND P. TAYLOR, *Predicting Overflow in an Emergency Department*, IMA Journal of Management Mathematics, **20** (2009), 39–49.

[3] C. BAIN, P. TAYLOR, G. MCDONNELL AND A. GEORGIOU, *Myths of ideal hospital occupancy*, Medical Journal of Australia, **192** (2010), 42–43.

[4] S. BROOKS AND A. GELMAN AND G. L. JONES AND X. MENG, *Handbook of Markov Chain Monte Carlo*, Chapman & Hall, New York, (2011).

[5] L. CAGGIANI AND M. OTTOMANELLI, *A dynamic simulation based model for optimal fleet repositioning in bike-sharing systems*, Procedia – Social and Behavioral Sciences, **87** (2013), 203–210.

[6] G. CASELLA AND C. ROBERT, *Monte Carlo Statistical Methods*, Springer Verlag, New York, (2004).

[7] R. R. COIFMAN AND M. MAGGIONI, *Diffusion wavelets*, Applied and Computational harmonic Analysis, **21** (2006), 53–94.

[8] E. CÔME AND L. OUKHRLLOU, *Model-Based Count Series Clustering for Bike Sharing System Usage Mining: A Case Study with the Vélib' System of Paris*, ACM Transactions on Intelligent Systems and Technology, **5** (3), (2014), 39:1–39:21.

[9] T. H. CORMEN AND C. E. LEISERSON AND R. L. RIVEST AND C. STEIN (Eds), *Introduction to Algorithms*, The MIT Press

[10] P. L'ECUYER AND C. LECOT AND B. TUFFIN, *A Randomized quasi-Monte Carlo Simulation Method for Markov Chains*, Operations Research **56** (2008), 958–975.

[11] P. FEARNHEAD AND C. SHERLOCK, *An exact Gibbs sampler for the Markov-modulated Poisson process*, J. R. Statist. Soc., **B 68** (2006), 767–784.

[12] W. R. GILKS (Eds), *Markov chain Monte Carlo in practice*, Chapman & Hall, London, (1996).

[13] P. GLASSERMAN (Eds), *Monte Carlo Methods in Financial Engineering*, Springer, New York, (2003).

[14] M. J. GOLDBERG AND S. KIM, *Applications of some formulas for finite Markov chains*, Applied and Computational harmonic Analysis, **30** (2011), 37–46.

[15] J. GUO, Y. WANG, K. S. TANG, S. CHAN, E. W. M. WONG, P. TAYLOR AND M. ZUKERMAN, *Evolutionary optimization of file assignment for a large-scale video-on-demand system*, IEEE Transactions on Knowledge and Data Engineering, **20** (2008), 836–850.

[16] J. GUO, E. W. M. WONG, S. CHAN, P. TAYLOR, M. ZUKERMAN AND K. S. TANG, *Combination load balancing for video-on-demand systems*, IEEE Transactions on Circuits and Systems for Video Technology, **18** (2008), 937–948.

[17] M. J. HASTINGS, *Monte Carlo sampling methods using Markov chains and their applications*, Biometrika, **57** (1970), 97–109.

[18] A. HOBOLTH, *A Markov chain Monte Carlo expectation maximization algorithm for statistical analysis of DNA sequence evolution with neighbour-dependent substitution rates*, Journal of Computational and Graphical Statistics, **17** (2008), 138–164.

[19] A. HOBOLTH AND E. A. STONE, *Efficient simulation from finite-state, continuous-time Markov chains with incomplete observations*, Ann. Appl. Statist., **3** (2009), 1204–1231.

[20] A. J. KLEYWEGTT, A. SHAPIRO AND T. HOMEM-DE-MELLO, *Society for Industrial and Applied Mathematics*, In: Technical report, Schuijbroek, **12** (2) (2001), 479–502.

[21] V. G. KULKARNI, *Introduction to Modeling and Analysis of Stochastic systems*, Springer, New York, (2011).

[22] H. NIEDERREITER (Eds), *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Philadelphia, (1992).

[23] J. SCHUIJBROEK, R. HAMPSHIRE AND W. J. VAN HOEVE, *Inventory rebalancing and vehicle routing in bike sharing systems*, In: Technical report, Schuijbroek, `http://repository.cmu.edu/tepper/1491/`, (Accessed September 2014), (2013).

(Received October 21, 2014)

Antonija Pršlja
Ljubljana 1000, Slovenia
e-mail: antonija.prslja@gmail.com