Computing 2-Step Transition Probabilities for Interval Markov Chains

Marcilia Andrade Campos[1], Graçaliz Pereira Dimuro[2], Antônio Carlos da Rocha Costa[2], and Vladik Kreinovich[3]

[1]Centro de Informática, Universidade Federal de Pernambuco
50670-901, Recife, Brazil, mac@cin.ufpe.br
[2]Escola de Informática, Universidade Católica de Pelotas
Rua Felix da Cunha 412, 96010-000, Pelotas, Brazil
liz@atlas.ucpel.tche.br, rocha@atlas.ucpel.tche.br
[3]Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA, vladik@cs.utep.edu

Abstract

Markov chains are a useful tool for solving practical problems. One of the useful features of a Markov chain is that once we know the 1-step transition probabilities – i.e., probabilities of a transition in a single step – we can easily compute 2-step probabilities. In many real-life situations, we do not know the exact values of transition probabilities; instead, we only know the intervals of possible values of these 1-step transition probabilities. In such situations, we must be able to compute the interval of possible values of 2-step transition probabilities. For this problem, intervals produced by straightforward interval computations have excess width. In this paper, we propose a new algorithm that computes the exact intervals for 2-step transition probabilities and that requires (asymptotically) the same computation time as the corresponding non-interval computations.

1 Introduction

What are Markov chains and why they are useful in applications. Many real-life systems ranging from weather to hardware to psychological systems randomly switch from one state to another. In many such systems, the transition probability does not depend on the history, only on the current state. In other words, the probability that a system will be in a state s by time t

depends only on its state at time $t-1$ and does not depend on its states at previous moments of time $t-2, \ldots$ Such random systems are called *Markov chains*; see, e.g., [2, 3, 8, 13, 14, 15, 16].

How to describe a Markov chain. At any given moment of time, with given measuring instruments at our disposal, we can only distinguish between finitely many states of a system. Therefore, for practical applications, it is sufficient to describe Markov chains with finitely many states. In such a system, we can enumerate the states into a sequence $s_1, \ldots, s_n$; then, to describe a Markov chain, it is sufficient to present transition probabilities p_{ij} of going from i-th state s_i in the previous moment of time to j-th state s_j in the next moment of time:

$$p_{ij} \stackrel{\text{def}}{=} P(s(t) = s_j \,|\, s(t-1) = s_i). \tag{1}$$

Once we are in a state s_i at a moment t, we must be in some state in the next moment of time; thus, for each original state, the sum of the corresponding transition probabilities should be equal to 1:

$$\sum_{j=1}^{n} p_{ij} = 1. \tag{2}$$

Possibility to easily compute 2-step transition probabilities: an important feature of Markov chains. One of the features that make Markov chains so useful in practice is that – due to the fact that the transition probabilities do not depend on the pre-history – once we know the 1-step transition probabilities (1), we can easily determine 2-step and other transition probabilities. For example, to determine the two-step probabilities

$$p_{ij}^{(2)} \stackrel{\text{def}}{=} P(s(t) = s_j \,|\, s(t-2) = s_i), \tag{3}$$

we can use the following formula:

$$p_{ij}^{(2)} = \sum_{k=1}^{n} p_{ik} \cdot p_{kj}. \tag{4}$$

How long does it take to compute these 2-step transition probabilities? Usually, software systems simply implement the formula (4). In this case, to compute each of n^2 values $p_{ij}^{(2)}$, we need n multiplications and $n-1$ additions; hence, the total number of arithmetic operations is equal to $n^2 \cdot (n + (n-1)) = O(n^3)$.

It is worth mentioning that for large n, we can, in principle, compute the 2-step transition probabilities faster than in $O(n^3)$ time: indeed, the formula (4) simply means that we multiply a matrix p_{ij} by itself, and it is known (see, e.g., [1] and references therein) that there are algorithms that multiply two matrices in time $O(n^\alpha)$ for $\alpha < 3$; however, these algorithms are rarely practically used

for Markov chains because they becomes more efficient than straightforward formula (4) only for very large n, and in most practical applications of Markov chains, we need chains with $n \ll 100$ states; see, e.g., [14].

Interval-valued Markov chains. Traditional Markov chain methods are based on the assumption that we know the exact values of the transition probabilities p_{ij}. In most practical situations, we do not know the exact values of the probabilities; at best, for each probability p_{ij}, we know a lower bound $\underline{p}_{ij}$ and an upper bound $\overline{p}_{ij}$. In other words, for every i and j, we only know the interval $\mathbf{p}_{ij} = [\underline{p}_{ij}, \overline{p}_{ij}]$ of possible values of p_{ij}; see, e.g., [10, 17, 18]. Such interval-valued Markov chains were introduced and analyzed in [9].

Problem: how to compute 2-step transition probabilities for interval Markov chains. We have already mentioned that computing 2-step transition probabilities is an important part of Markov chain applications. Thus, when we only know intervals of possible values of p_{ij}, we must be able to compute intervals of possible values of 2-step transition probabilities.

At first glance, this problem may seem straightforward: we have an explicit formula (4) for computing the desired values $p_{ij}^{(2)}$ from p_{ij}, so we can simply use straightforward interval computation (see, e.g., [5, 6, 7, 12]), i.e., perform the same operations on intervals $\mathbf{p}_{ij}$ instead of numbers. As a result, we get enclosures for the desired intervals $\mathbf{p}_{ij}^{(2)}$.

The problem with this approach is that we do not take into consideration that the values p_{ij} cannot independently take arbitrary values within the corresponding intervals $\mathbf{p}_{ij}$: these values must satisfy the additional constraint (2). Since we did not take these constraints into consideration, the results of straightforward interval computations often have excess width. The possibility of such an excess width can be easily illustrated on a toy example when we have no information about the transition probabilities at all, i.e., when for each i and j, we have $\mathbf{p}_{ij} = [0,1]$. In this case, straightforward interval computations lead to $\mathbf{p}_{ik} \cdot \mathbf{p}_{kj} = [0,1] \cdot [0,1] = [0,1]$ hence, for the sum $p_{ij}^{(2)}$ of n such terms, we get an interval $[0,n]$. Since the actual probability has to be within the interval $[0,1]$, we clearly have excess width here.

What we are planning to do. Since we cannot compute the exact range of $\mathbf{p}_{ij}^{(2)}$ by using straightforward interval computations, we must design new techniques. In this paper, we propose a new algorithm for computing the exact interval range for 2-step transition probabilities, and we show that this algorithms enables us to compute these ranges in exactly the same asymptotic time $O(n^3)$ as for number-valued Markov chains.

2 Analysis of the Problem and a Step-By-Step Design of an Efficient Algorithm for Solving This Problem

Reduction to SUE expressions. In interval computations, one known source of excess width is repetition of variables. It is known that if a formula is a *single-use expression* (SUE), i.e., if in this formula, each variable only occurs once, that for such formulas, straightforward interval computations lead to the exact range (see, e.g., [4]). To avoid this excess width, let us first represent the expression (4) in SUE form.

The original formulas have few repetitions of variables, so this reduction can be easily done. The resulting expressions are different for $i = j$ and for $i \neq j$. For $i = j$, we get the following SUE expression:

$$p_{ii}^{(2)} = \sum_{k \neq i} p_{ik} \cdot p_{kj} + p_{ii}^2. \tag{5}$$

For $i \neq j$, we get the following SUE expression:

$$p_{ij}^{(2)} = \sum_{k \neq i,j} p_{ik} \cdot p_{kj} + p_{ij} \cdot (p_{ii} + p_{jj}). \tag{6}$$

Peeling produces the exact range but requires lots of computations. To compute the exact range of $p_{ij}^{(2)}$, we must find the maximum and the minimum of the corresponding expressions (5) and (6) under the conditions (2) and

$$\underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij}. \tag{7}$$

In other words, we want to optimize a quadratic function under linear constraints (equalities and inequalities). In principle, to optimize such a function, we can use the idea of peeling; see, e.g, [6]. For our problem, the idea of peeling can be described as follows.

We want to optimize a quadratic function over the region described by linear equalities and inequalities (2) and (7). From the geometric viewpoint, a region described by linear equalities and inequalities is a polytope. The maximum (or a minimum) of a function in this region is attained either in the interior of this polytope, or in one of its lower-dimensional boundary polyhedral elements: faces, faces of the faces, ..., all the way to 0-dimensional elements – vertices.

Based on the equalities and inequalities that describe a polytope, we can explicitly describe all these polyhedral elements; there are $\approx 2^n$ of them. For each of these boundary elements, we can select independent variables $x_{i_1}, \ldots, x_{i_d}$ – as many as the dimension d of this boundary element – and explicitly describe other variables x_i as linear functions of these independent ones. (In the limit case, when we consider vertices – 0-dimensional boundary elements – there are

no independent variables at all.) If we substitute the expressions for all the variables in terms of independent ones into the optimized quadratic function, then we get an expression $E(x_{i_1}, \ldots, x_{i_d})$ for this quadratic function in terms of d independent variables $x_{i_1}, \ldots, x_{i_d}$ only. If the minimum or maximum of this expression is in the interior of the boundary element, then all d partial derivatives w.r.t. these variables should be equal to 0:

$$\frac{\partial E}{\partial x_{i_k}} = 0. \tag{8}$$

Since the derivative of a quadratic function is a linear function, the equations (8) forms a system of d linear equations with d unknowns $x_{i_1}, \ldots, x_{i_d}$. This system is easy to solve, so for each boundary element, we get a possible optimum point. Of course, we also need to check that this solution does belong to this boundary element (and not to its extension) by checking that all linear inequalities that define this element are satisfied. (For vertices, we simply compute the value of the quadratic function.)

Summarizing: we know that, e.g., the minimum is always attained either in the interior of the polytope, or in the interior of one of the boundary elements, and for each boundary element, we know how to compute the point where this minimum is attained (if at all). Thus, to find the minimum of the given quadratic function, we simply analyze all the boundary elements this way, and then take the smallest of the corresponding values of the minimized quadratic function.

This algorithm enables us to compute the exact range of any quadratic function, in particular, of the function $p_{ij}^{(2)}$. The only problem with this algorithm – as mentioned in [6] – is that since we have $\approx 2^n$ boundary elements, this algorithm requires exponential ($\approx 2^n$) time.

So, if we want to compute the range in polynomial time, we must design a new algorithm. Our new algorithm will actually use peeling – but not peeling applied to the original problem, but peeling applied to reduced problems (with fewer variables).

Reduction to a fewer-dimensional problem: Step 1. Let us describe how this reduction can be done.

We will start with the case when $i = j$ and we are looking for the maximum of the quadratic expression (5). In this case, we want to solve the following problem:

$$\sum_{k \neq i} p_{ik} \cdot p_{ki} + p_{ii}^2 \to \max \tag{9}$$

under the conditions that $p_{ab} \in \mathbf{p}_{ab}$ for all a and b and that

$$\sum_{k \neq i} p_{ik} + p_{ii} = 1. \tag{10}$$

In (9), the coefficients at p_{ki} are non-negative; therefore, the maximum is attained when each of the terms p_{ki} attains the largest possible value $\overline{p}_{ki}$. In other words, the solution to the problem (9) is also a solution to the following problem with fewer unknowns:

$$\sum_{k \neq i} p_{ik} \cdot \overline{p}_{ki} + p_{ii}^2 \to \max \tag{11}$$

under the conditions (10) and

$$\underline{p}_{ik} \leq p_{ik} \leq \overline{p}_{ik}. \tag{12}$$

Reduction to a fewer-dimensional problem: Step 2. Let p_{ii} be the value for which the maximum is attained. Then, if we fix the value p_{ii}, we get the following problem with one fewer variable:

$$\sum_{k \neq i} p_{ik} \cdot \overline{p}_{ki} \to \max \tag{13}$$

under the conditions (12) and

$$\sum_{k \neq i} p_{ik} = 1 - p_{ii}. \tag{14}$$

Reduction to a fewer-dimensional problem: Step 3. To perform a further reduction, let us sort that the coefficients $\overline{p}_{ki}$ $(k \neq i)$ in decreasing order, i.e., in such a way that

$$\overline{p}_{(1)i} \geq \overline{p}_{(2)i} \geq \cdots \geq \overline{p}_{(n-1)i}. \tag{15}$$

The sums in (13) and (14) do not depend on the order in which we add the terms. Thus, the above optimization problem can be reformulated as follows:

$$\sum_{k} p_{i(k)} \cdot \overline{p}_{(k)i} \to \max \tag{16}$$

(where $p_{i(k)}$ denotes the value p_{il} for which $\overline{p}_{li} = \overline{p}_{(k)i}$) under the conditions

$$\underline{p}_{i(k)} \leq p_{i(k)} \leq \overline{p}_{i(k)} \tag{17}$$

and

$$\sum_{k} p_{i(k)} = 1 - p_{ii}. \tag{18}$$

In this case, if, for some $k_1 < k_2$, we have $p_{i(k_1)} < \overline{p}_{i(k_1)}$ and $p_{i(k_2)} > \underline{p}_{i(k_2)}$, then we can subtract a small positive value $\varepsilon > 0$ from $p_{i(k_2)}$ and add this value to $p_{i(k_1)}$, i.e., replace $p_{i(k_1)}$ with $p'_{i(k_1)} = p_{i(k_1)} + \varepsilon$ and $p'_{i(k_2)} = p_{i(k_2)} - \varepsilon$ (we keep all other values $p_{i(k)}$ unchanged). If ε is small enough, we still satisfy the

conditions $p_{i(k_1)} \in \mathbf{p}_{i(k_1)}$ and $p_{i(k_2)} \in \mathbf{p}_{i(k_2)}$. Since we added and subtracted the same value, the sum of the resulting probabilities remains the same hence, the condition (18) is still satisfied, so the new values $p'_{i(k)}$ satisfy all the necessary conditions.

If we replace $p_{i(k)}$ by $p'_{i(k)}$, then the value of the optimized function (16) is increased by $\varepsilon \cdot (\overline{p}_{(k_1)i} - \overline{p}_{(k_2)i})$. Since the values $\overline{p}_{(k)i}$ are sorted in decreasing order, and $k_1 < k_2$, we conclude that the increase is non-negative. Thus, if $p_{i(k_1)} < \overline{p}_{i(k_1)}$ and $p_{i(k_2)} > \underline{p}_{i(k_2)}$, we can change the values of $p_{i(k)}$ in such a way that one of these conditions is no longer true, and increase (or at least not decrease) the value of the optimized function.

Hence, a maximum is attained at a vector $(p_{i(1)}, p_{i(2)}, \ldots)$ for which the above condition is never satisfied, i.e., for which:

- once $p_{i(k')} < \overline{p}_{i(k')}$, we have $p_{i(k)} = \underline{p}_{i(k)}$ for all $k > k'$, and
- once $p_{i(k')} > \underline{p}_{i(k')}$, we have $p_{i(k)} = \overline{p}_{i(k)}$ for all $k < k'$.

Thus, if there is a k' for which $\underline{p}_{i(k')} < p_{i(k')} < \overline{p}_{i(k')}$, we have $p_{i(k)} = \overline{p}_{i(k)}$ for all $k < k'$ and $p_{i(k)} = \underline{p}_{i(k)}$ for all $k > k'$.

If there is no such k', i.e., if for every k, $p_{i(k)} = \underline{p}_{i(k)}$ or $p_{i(k)} = \overline{p}_{i(k)}$, then once $p_{i(k)} = \overline{p}_{i(k)}$ and hence $p_{i(k)} > \underline{p}_{i(k)}$, we have $p_{i(k')} = \overline{p}_{i(k')}$ for all $k' < k$. Similarly, once $p_{i(k)} = \underline{p}_{i(k)}$ and hence $p_{i(k)} < \overline{p}_{i(k)}$, we have $p_{i(k')} = \underline{p}_{i(k')}$ for all $k' > k$.

In all these cases, there is a borderline value k' such that $p_{i(k)} = \overline{p}_{i(k)}$ for all $k < k'$ and $p_{i(k)} = \underline{p}_{i(k)}$ for all $k > k'$. Thus, once we fixed k', the optimal values of all the variables $p_{i(k)}$ are fixed except for one variable: $p_{i(k')}$. Hence, once k' is fixed, the original optimization problem takes the following form:

$$p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ii}^2 \to \max \tag{19}$$

under the conditions that

$$\underline{p}_{i(k')} \le p_{i(k')} \le \overline{p}_{i(k')}, \tag{20}$$

$$\underline{p}_{ii} \le p_{ii} \le \overline{p}_{ii}, \tag{21}$$

and

$$p_{i(k')} + p_{ii} = c_{k'}, \tag{22}$$

where we denoted

$$c_{k'} \stackrel{\text{def}}{=} 1 - \sum_{k:k<k'} \overline{p}_{i(k)} - \sum_{k:k>k'} \underline{p}_{i(k)}. \tag{23}$$

This is a quadratic optimization problem with 2 variables under linear constraints, so, for this problem, the peeling method leads to a solution in $2^2 = \text{const}$ number of computational steps.

Once we compute the optimal values of $p_{i(k')}$ and p_{ii}, we can compute the corresponding value $V_{k'}$ of the original objective function as

$$V_{k'} = \sum_{k:k<k'} \overline{p}_{i(k)} \cdot \overline{p}_{(k)i} + p_{i(k')} \cdot \overline{p}_{(k')i} + \sum_{k:k>k'} \underline{p}_{i(k)} \cdot \overline{p}_{(k)i} + p_{ii}^2. \qquad (24)$$

The actual maximum of $p_{ii}^{(2)}$ can then be determined as the largest of the corresponding values $V_{k'}$.

Resulting algorithm for computing the upper bound for $p_{ii}^{(2)}$: first draft. The above analysis leads to the following algorithm for computing, for a given i, the upper endpoint $\overline{p_{ii}^{(2)}}$ of the interval of possible values of $p_{ii}^{(2)}$:

- First, we sort the values $\overline{p}_{ki}$ $(k \neq i)$ in decreasing order: $\overline{p}_{(1)i} \geq \overline{p}_{(2)i} \geq \cdots \geq \overline{p}_{(n-1)i}$.
- Then, for each $k' = 1, \ldots, n-1$, we do the following:
  - we compute the value $c_{k'}$ by using formula (23);
  - we solve the problem (19)–(22) of optimizing a quadratic function of two variables $p_{i(k')}$ and p_{ii} with linear constraints;
  - based on the solution, we compute $V_{k'}$ by using the formula (24).
- Finally, we return the largest of the values $V_1, \ldots, V_{n-1}$ as the solution to the original optimization problem.

What is the computational complexity of this algorithm? Sorting requires $O(n \cdot \log(n))$ steps (see, e.g., [1]). After sorting, for each k', we need:

- $O(n)$ steps to compute the sums in $c_{k'}$,
- then a constant number of steps to solve the optimization problem with 2 unknowns, and
- then, again $O(n)$ steps to compute $O(n)$ steps –

the total of $O(n)$ steps. Since we need $O(n)$ steps of each of $n-1$ values k', we thus need a total of $O(n^2)$ steps. The final computation of the largest of $n-1$ values $V_{k'}$ requires $O(n)$ steps, so the overall computational complexity of the after-sorting part of this algorithm is $O(n^2) + O(n \cdot \log(n)) = O(n^2)$.

This is much larger than $O(n)$ steps that is necessary to compute the value of $p_{ii}^{(2)}$ in the non-interval case, by using the formula (4). It is therefore desirable to decrease the computation time of our algorithm. How can we do that?

Decreasing the computation time of the resulting algorithm. It is indeed possible to reduce the above computation time because we do not really need to compute $c_{k'}$ and $V_{k'}$ "from scratch" every time: for each $k' > 1$, we can compute the values of these variables by modifying the previous values. More specifically, we can compute the auxiliary values

$$W_{k'} \stackrel{\text{def}}{=} \sum_{k:k<k'} \overline{p}_{i(k)} \cdot \overline{p}_{(k)i} + \sum_{k:k>k'} \underline{p}_{i(k)} \cdot \overline{p}_{(k)i}, \tag{25}$$

for which

$$V_{k'} = W_{k'} + p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ii}^2. \tag{26}$$

To be more precise, we do need to compute the original values c_1 and W_1; they will be computed as

$$c_1 = 1 - \sum_{k:k>1} \underline{p}_{i(k)} \tag{27}$$

and

$$W_1 = \sum_{k:k>1} \underline{p}_{i(k)} \cdot \overline{p}_{(k)i}. \tag{28}$$

After that, once we know the values c_{k-1} and W_{k-1}, we can compute the next values of c_k and W_k by using the following easy-to-deduce formulas:

$$c_k = c_{k-1} - \overline{p}_{i(k-1)} + \underline{p}_{i(k)} \tag{29}$$

and

$$W_k = W_{k-1} + \overline{p}_{i(k-1)} \cdot \overline{p}_{(k-1)i} - \underline{p}_{i(k)} \cdot \overline{p}_{(k)i}. \tag{30}$$

After this modification, the after-sorting part of the algorithm requires that for each $k' = 1, \ldots, n-1$, we do the following:

- first, we compute the value $c_{k'}$; for $k' = 1$, we use the formula (27); for $k' > 1$, we use the formula (29);
- we solve the problem (19)–(22) of optimizing a quadratic function of two variables $p_{i(k')}$ and p_{ii} with linear constraints;
- we compute the value $W_{k'}$; for $k' = 1$, we use the formula (28); for $k' > 1$, we use the formula (30);
- based on the solution of the quadratic optimization problem, we compute $V_{k'}$ by using the formula (26).

Here, for $k' = 1$, we need $O(n)$ steps, but for every other k', we only need finitely many steps. Thus, the overall after-sorting complexity of this algorithm is $O(n)$ – exactly the same as in the non-interval case.

Similar algorithm for computing the upper bound for $p_{ij}^{(2)}$ $(j \neq i)$. For the case $j \neq i$, similar reductions, when applied to the formula (6), lead to the conclusion that the desired upper endpoints is a solution to the following simplified optimization problem:

$$\sum_{k \neq i,j} p_{ik} \cdot \overline{p}_{kj} + p_{ij} \cdot (p_{ii} + \overline{p}_{jj}) \to \max \tag{31}$$

under the conditions that $p_{ab} \in \mathbf{p}_{ab}$ for all a and b and that

$$\sum_{k \neq i,j} p_{ik} + p_{ii} + p_{ij} = 1. \tag{32}$$

After sorting the values $\overline{p}_{kj}$ $(k \neq i, j)$ in decreasing order, we can prove that there exists a borderline value k' for which $p_{i(k)} = \overline{p}_{i(k)}$ for all $k < k'$, $p_{i(k)} = \underline{p}_{i(k)}$ for all $k > k'$, and the values $p_{i(k')}$, p_{ii}, and p_{ij} can be obtained by solving the following quadratic optimization problem with linear constraints:

$$p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ij} \cdot (p_{ii} + \overline{p}_{jj}) \to \max \tag{33}$$

under the conditions that

$$\underline{p}_{i(k')} \leq p_{i(k')} \leq \overline{p}_{i(k')}, \tag{34}$$

$$\underline{p}_{ii} \leq p_{ii} \leq \overline{p}_{ii}, \tag{35}$$

$$\underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij}, \tag{36}$$

and

$$p_{i(k')} + p_{ii} + p_{ij} = c_{k'}, \tag{37}$$

where we denoted

$$c_{k'} \stackrel{\text{def}}{=} 1 - \sum_{k:k<k'} \overline{p}_{i(k)} - \sum_{k:k>k'} \underline{p}_{i(k)}. \tag{38}$$

This is a quadratic optimization problem with 3 variables under linear constraints, so, for this problem, the peeling method leads to a solution in $2^3 = \text{const}$ number of computational steps.

The above expression for the objective function can be reformulated as

$$V_{k'} = W_{k'} + p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ij} \cdot (p_{ii} + \overline{p}_{jj}). \tag{39}$$

Thus, similarly to the case $i = j$, we can set up the after-sorting part of our algorithm as doing the following for each $k' = 1, \ldots, n-2$:

- first, we compute the value $c_{k'}$; for $k' = 1$, we use the formula (27); for $k' > 1$, we use the formula (29);

- we solve the problem (33)–(37) of optimizing a quadratic function of three variables $p_{i(k')}$, p_{ii}, and p_{ij} with linear constraints;
- we compute the auxiliary value $W_{k'}$; for $k' = 1$, we use the formula (28); for $k' > 1$, we use the formula (30);
- based on the solution of the quadratic optimization problem, we compute $V_{k'}$ by using the formula (39).

Here, for $k' = 1$, we need $O(n)$ steps, but for every other k', we only need finitely many steps. Thus, the overall after-sorting complexity of this algorithm is $O(n)$ – exactly the same as in the non-interval case.

Overall computational complexity. Overall, we need to sort n sequences corresponding to n different values of i. Thus, all the sorting requires

$$n \cdot O(n \cdot \log(n)) = O(n^2 \cdot \log(n)) \ll O(n^3) \tag{40}$$

steps. After sorting, we need $O(n)$ steps to compute each of n^2 upper bounds; therefore, we need $O(n^3)$ after-sorting steps. Overall, the above algorithm requires $O(n^2 \cdot \log(n)) + O(n^3) = O(n^3)$ steps – asymptotically the same number of steps as in the non-interval case.

Similar algorithm for computing the lower bound for $p_{ij}^{(2)}$. To compute the lower endpoint for $p_{ij}^{(2)}$, we must use $\underline{p}_{ki}$ instead of $\overline{p}_{ki}$; therefore, we must sort the values $\underline{p}_{ki}$ instead of $\overline{p}_{ki}$, and we must solve the corresponding minimization problems instead of the maximization ones.

We are now ready to describe the algorithms. In the following two sections, we will now summarize the resulting $O(n^3)$ algorithms for computing 2-step transition probabilities for interval Markov chains.

3 $O(n^3)$ Algorithm for Computing the Exact Upper Bound for 2-Step Transition Probabilities for Interval Markov Chains

First part. First, for each i from 1 to n, we sort the values $\overline{p}_{1i}, \ldots, \overline{p}_{ni}$ into a decreasing sequence.

Second part. Then, for each i from 1 to n, to compute the exact upper bound $\overline{p_{ii}^{(2)}}$ for $p_{ii}^{(2)}$, we do the following. First, by deleting the value $\overline{p}_{ii}$ from the sorted sequence, we get a sorting of all the values $\overline{p}_{ki}$ $(k \neq i)$ into a decreasing sequence

$$\overline{p}_{(1)i} \geq \overline{p}_{(2)i} \geq \cdots \geq \overline{p}_{(n-1)i}. \tag{41}$$

Then, for each $k' = 1, 2, \ldots, n-1$, we do the following:

- first, we compute the value $c_{k'}$; for $k' = 1$, we use the formula

$$c_1 = 1 - \sum_{k:k>1} \underline{p}_{i(k)}; \tag{42}$$

  for $k' > 1$, we use the formula

$$c_{k'} = c_{k'-1} - \overline{p}_{i(k'-1)} + \underline{p}_{i(k')}; \tag{43}$$

- we use peeling to solve the problem of optimizing a quadratic function

$$p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ii}^2 \to \max \tag{44}$$

  of two variables $p_{i(k')}$ and p_{ii} under linear conditions that

$$\underline{p}_{i(k')} \le p_{i(k')} \le \overline{p}_{i(k')}, \tag{45}$$

$$\underline{p}_{ii} \le p_{ii} \le \overline{p}_{ii}, \tag{46}$$

  and

$$p_{i(k')} + p_{ii} = c_{k'}; \tag{47}$$

- we compute the value $W_{k'}$; for $k' = 1$, we use the formula

$$W_1 = \sum_{k:k>1} \underline{p}_{i(k)} \cdot \overline{p}_{(k)i}; \tag{48}$$

  for $k' > 1$, we use the formula

$$W_{k'} = W_{k'-1} + \overline{p}_{i(k'-1)} \cdot \overline{p}_{(k'-1)i} - \underline{p}_{i(k')} \cdot \overline{p}_{(k')i}; \tag{49}$$

- based on the solution of the quadratic optimization problem, we compute $V_{k'}$ by using the formula

$$V_{k'} = W_{k'} + p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ii}^2. \tag{50}$$

The actual maximum of $p_{ii}^{(2)}$ can then be determined as the largest of the corresponding values $V_{k'}$.

Third part. Finally, for each i from 1 to n and for each j from 1 to n, to compute the exact upper bound $\overline{p_{ij}^{(2)}}$ for $p_{ij}^{(2)}$, we do the following. First, by deleting the values $\overline{p}_{ii}$ and $\overline{p}_{ij}$ from the sorted sequence, we get a sorting of all the values $\overline{p}_{ki}$ $(k \neq i, j)$ into a decreasing sequence

$$\overline{p}_{(1)i} \geq \overline{p}_{(2)i} \geq \cdots \geq \overline{p}_{(n-2)i}. \tag{51}$$

Then, for each $k' = 1, 2, \ldots, n-1$, we do the following:

- first, we compute the value $c_{k'}$; for $k' = 1$, we use the formula (42); for $k' > 1$, we use the formula (43);
- we use peeling to solve the problem of optimizing a quadratic function

  $$p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ij} \cdot (p_{ii} + \overline{p}_{jj}) \to \max \tag{52}$$

  of three variables $p_{i(k')}$, p_{ii}, and p_{ij} under linear conditions that

  $$\underline{p}_{i(k')} \leq p_{i(k')} \leq \overline{p}_{i(k')}, \tag{53}$$

  $$\underline{p}_{ii} \leq p_{ii} \leq \overline{p}_{ii}, \tag{54}$$

  $$\underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij}, \tag{55}$$

  and

  $$p_{i(k')} + p_{ii} + p_{ij} = c_{k'}; \tag{56}$$

- we compute the value $W_{k'}$; for $k' = 1$, we use the formula (48); for $k' > 1$, we use the formula (49);
- based on the solution of the quadratic optimization problem, we compute $V_{k'}$ by using the formula

  $$V_{k'} = W_{k'} + p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ij} \cdot (p_{ii} + \overline{p}_{jj}). \tag{57}$$

The actual maximum of $p_{ij}^{(2)}$ can then be determined as the largest of the corresponding values $V_{k'}$.

4 $O(n^3)$ Algorithm for Computing the Exact Lower Bound for 2-Step Transition Probabilities for Interval Markov Chains

First part. First, for each i from 1 to n, we sort the values $\underline{p}_{1i}, \ldots, \underline{p}_{ni}$ into a decreasing sequence.

Second part. Then, for each i from 1 to n, to compute the exact lower bound $\underline{p_{ii}^{(2)}}$ for $p_{ii}^{(2)}$, we do the following. First, by deleting the value $\underline{p}_{ii}$ from the sorted sequence, we get a sorting of all the values $\underline{p}_{ki}$ $(k \neq i)$ into a decreasing sequence

$$\underline{p}_{(1)i} \geq \underline{p}_{(2)i} \geq \cdots \geq \underline{p}_{(n-1)i}. \tag{58}$$

Then, for each $k' = 1, 2, \ldots, n-1$, we do the following:

- first, we compute the value $c_{k'}$; for $k' = 1$, we use the formula

$$c_1 = 1 - \sum_{k:k>1} \overline{p}_{i(k)}; \tag{59}$$

  for $k' > 1$, we use the formula

$$c_{k'} = c_{k'-1} - \underline{p}_{i(k'-1)} + \overline{p}_{i(k')}; \tag{60}$$

- we use peeling to solve the problem of optimizing a quadratic function

$$p_{i(k')} \cdot \underline{p}_{(k')i} + p_{ii}^2 \to \min \tag{61}$$

  of two variables $p_{i(k')}$ and p_{ii} under linear conditions that

$$\underline{p}_{i(k')} \leq p_{i(k')} \leq \overline{p}_{i(k')}, \tag{62}$$

$$\underline{p}_{ii} \leq p_{ii} \leq \overline{p}_{ii}, \tag{63}$$

  and

$$p_{i(k')} + p_{ii} = c_{k'}; \tag{64}$$

- we compute the value $W_{k'}$; for $k' = 1$, we use the formula

$$W_1 = \sum_{k:k>1} \overline{p}_{i(k)} \cdot \underline{p}_{(k)i}; \tag{65}$$

  for $k' > 1$, we use the formula

$$W_{k'} = W_{k'-1} + \underline{p}_{i(k'-1)} \cdot \underline{p}_{(k'-1)i} - \overline{p}_{i(k')} \cdot \underline{p}_{(k')i}; \tag{66}$$

- based on the solution of the quadratic optimization problem, we compute $V_{k'}$ by using the formula

$$V_{k'} = W_{k'} + p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ii}^2. \tag{67}$$

The actual minimum of $p_{ii}^{(2)}$ can then be determined as the smallest of the corresponding values $V_{k'}$.

Third part. Finally, for each i from 1 to n and for each j from 1 to n, to compute the exact lower bound $\underline{p}_{ij}^{(2)}$ for $p_{ij}^{(2)}$, we do the following. First, by deleting the values $\underline{p}_{ii}$ and $\underline{p}_{ij}$ from the sorted sequence, we get a sorting of all the values $\underline{p}_{ki}$ $(k \neq i, j)$ into a decreasing sequence

$$\underline{p}_{(1)i} \geq \underline{p}_{(2)i} \geq \cdots \geq \underline{p}_{(n-2)i}. \tag{68}$$

Then, for each $k' = 1, 2, \ldots, n-1$, we do the following:

- first, we compute the value $c_{k'}$; for $k' = 1$, we use the formula (59); for $k' > 1$, we use the formula (60);
- we use peeling to solve the problem of optimizing a quadratic function

$$p_{i(k')} \cdot \underline{p}_{(k')i} + p_{ij} \cdot (p_{ii} + \underline{p}_{jj}) \to \min \tag{69}$$

  of three variables $p_{i(k')}$, p_{ii}, and p_{ij} under linear conditions that

$$\underline{p}_{i(k')} \leq p_{i(k')} \leq \overline{p}_{i(k')}, \tag{70}$$

$$\underline{p}_{ii} \leq p_{ii} \leq \overline{p}_{ii}, \tag{71}$$

$$\underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij}, \tag{72}$$

  and

$$p_{i(k')} + p_{ii} + p_{ij} = c_{k'}; \tag{73}$$

- we compute the value $W_{k'}$; for $k' = 1$, we use the formula (65); for $k' > 1$, we use the formula (66);
- based on the solution of the quadratic optimization problem, we compute $V_{k'}$ by using the formula

$$V_{k'} = W_{k'} + p_{i(k')} \cdot \underline{p}_{(k')i} + p_{ij} \cdot (p_{ii} + \underline{p}_{jj}). \tag{74}$$

The actual minimum of $p_{ij}^{(2)}$ can then be determined as the smallest of the corresponding values $V_{k'}$.

5 Final Comment: What About 3-Step Probabilities?

A natural question is: we know how to compute the exact intervals for 2-step transition probabilities, what about 3-step transition probabilities? 4-step probabilities?

How to compute these probabilities and whether it is even possible to compute the exact intervals for these probabilities in reasonable time, is an open question. The reason why it may be difficult is that in general, just like the formula (4) means that the matrix formed by 2-step probabilities is a square of the matrix p_{ij}, the matrix formed by 3-step probabilities is a cube of the original matrix p_{ij}. The square of a matrix can be reduced to a single-use expression and thus, computed exactly even in the interval case. The cube of a matrix is difficult to represent in SUE form and, in general, computing the exact cube of an interval matrix is NP-hard [11].

This NP-hardness of a general problem does not necessarily mean that the sub-problem of computing the cube of a probability matrix is necessarily NP-hard, so there is hope that computing the exact intervals for 3-step transition probabilities may also turn out to be feasible.

Acknowledgments

This work was partially supported by the Brazilian funding agencies CNPq/CTPETRO, CNPq/CTINFO, and FAPERGS.

V.K. was also partly supported by NASA under cooperative agreement NCC5-209, by Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-1-0365, by NSF grants EAR-0112968 and EAR-0225670, by the Army Research Laboratories grant DATM-05-02-C-0046, and by IEEE/ACM SC2003 Minority Serving Institutions Participation Grant.

This work was mainly done when Graçaliz Pereira Dimuro and Antônio Carlos da Rocha Costa visited El Paso, Texas; this visit was sponsored by CNPq/CTPETRO and CNPq/CTINFO.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, and Mc-Graw Hill Co., N.Y., 2001.

[2] D. Gamerman, *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*, CRC Press, Boca Raton, Florida, 1997.

[3] W. R. Gilks, S. Richardson, and D. L. Spiegelhalter (Eds.), *Markov Chain Monte Carlo in Practice*, Chapman & Hall, Boca Raton, Florida, 1996.

[4] E. Hansen, "Sharpness in interval computations", *Reliable Computing*, 1997, Vol. 3, pp. 7–29.

[5] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer, London, 2001.

[6] R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer, Dordrecht, 1996.

[7] R. B. Kearfott and V. Kreinovich (eds.), *Applications of Interval Computations*, Kluwer, Dordrecht, 1996.

[8] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*, Springer Verlag, N.Y., 1976.

[9] I. O. Kozine and L. V. Utkin, "Interval-valued finite Markov chains", *Reliable Computing*, 2002, Vol. 8, No. 2, pp. 97–113.

[10] V. P. Kuznetsov, *Interval Statistical Models*, Radio and Communications publ., Moscow, 1991 (in Russian).

[11] G. Mayer and V. Kreinovich, *Computing $A \cdot A \cdot A$ is NP-hard*, working paper.

[12] R. E. Moore, *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.

[13] A. Papoulis, "Brownian Movement and Markoff Processes." In: *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, 1984, pp. 515–553.

[14] L. R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition", *Proceedings of the IEEE*, 1989, Vol. 77, No. 2, pp. 257–286.

[15] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, New Jersey, 1995.

[16] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*, J. Wiley, New York, 2001.

[17] P. Walley, *Statistical Reasoning with Imprecise Probabilities*, Chapman & Hall, London, 1991.

[18] P. Walley, "Measures of uncertainty", *Artificial Intelligence*, 1996, Vol. 83, pp. 1–58.