4-2007

Under Interval and Fuzzy Uncertainty, Symmetric Markov Chains are More Difficult to Predict

Roberto Araiza
The University of Texas at El Paso, raraiza@miners.utep.edu

Gang Xiang

Olga Kosheleva
The University of Texas at El Paso, olgak@utep.edu

Damjan Skulj

Comments:
Technical Report: UTEP-CS-07-23
Published in: Marek Reformat and Michael R. Berthold (eds.), *Proceedings of the 26th International Conference of the North American Fuzzy Information Processing Society NAFIPS'2007*, San Diego, California, June 24-27, 2007, pp. 526-531.

Under Interval and Fuzzy Uncertainty, Symmetric Markov Chains Are More Difficult to Predict

Roberto Araiza, Gang Xiang
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
{raraiza,gxiang}@utep.edu

Olga Kosheleva
Department of Teacher Education
University of Texas at El Paso
El Paso, TX 79968, USA
olgak@utep.edu

Damjan Škulj
Faculty of Social Sciences
University of Ljubljana
100 Ljubljana, Slovenia
Damjan.Skulj@Fdv.Uni-Lj.Si

***Abstract*—Markov chains are an important tool for solving practical problems. In particular, Markov chains have been successfully applied in bioinformatics. Traditional statistical tools for processing Markov chains assume that we know the exact probabilities p_{ij} of a transition from the state i to the state j. In reality, we often only know these transition probabilities with interval (or fuzzy) uncertainty. We start the paper with a brief reminder of how the Markov chain formulas can be extended to the cases of such interval and fuzzy uncertainty.**

In some practical situations, there is another restriction on the Markov chain–that this Markov chain is symmetric in the sense that for every two states i and j, the probability of transitioning from i to j is the same as the probability of transitioning from j to i: $p_{ij} = p_{ji}$. In general, symmetry assumptions simplify computations. In this paper, we show that for Markov chains under interval and fuzzy uncertainty, symmetry has the opposite effect: it makes the computational problems more difficult.

I. Markov Chains are Important

Many real-life processes are described by *Markov chains*, in which the probability p_{ij} of going from state i to state j depends only on these two states and does not depend on the previous history. In particular, Markov chains are actively used to described gene-related processes in bioinformatics; see, e.g., [4], [12].

For each state i, the probabilities p_{ij} of going to different states $j = 1, \ldots, n$ should add up to one:

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

One computational advantage of Markov chains is that once we know the probabilities p_{ij} of one-step transitions, we can determine the probabilities $p_{ij}^{(2)}$ of 2-step transitions as

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

Similarly, we can then define the probabilities of 3-step transitions, etc.

II. Markov Chains under Interval and Fuzzy Uncertainty

In practice, we often do not know the exact values of the transition probabilities p_{ij}. Instead, we know the *intervals* $[\underline{p}_{ij}, \overline{p}_{ij}]$ of possible values of p_{ij}, or, even more generally, fuzzy numbers μ_{ij} which describe these probabilities; see, e.g., [13], [19].

A natural question is: once we have this information about the one-step transition probabilities, what can we conclude about the 2-step transition probabilities? For example, in the interval case, we would like to know the intervals

$$\mathbf{p}_{ij}^{(2)} = \left\{ \sum_{k=1}^{n} p_{ik} \cdot p_{kj} : p_{ab} \in \mathbf{p}_{ab}, \sum_{b=1}^{n} p_{ab} = 1 \right\}.$$

In the fuzzy case, we would like to know the fuzzy sets corresponding to $p_{ij}^{(2)}$.

III. From the Computational Viewpoint, It Is Sufficient to Consider Interval Uncertainty

In the fuzzy case, to describe the corresponding uncertainty, for each value p of the probability p_{ij}, we describe the degree $\mu_{ij}(p)$ to which this value is possible.

For each degree of certainty α, we can determine the set of values of p_{ij} that are possible with at least this degree of certainty – the *α-cut* $\mathbf{p}_{ij}(\alpha) \stackrel{\text{def}}{=} \{p \,|\, \mu_{ij}(p) \geq \alpha\}$ of the original fuzzy set. In many practical cases, this α-cut is an interval.

Vice versa, if we know α-cuts for every α, then, for each value p, we can determine the degree of possibility that p belongs to the original fuzzy set [1], [3], [10], [15], [16], [17]. A fuzzy set can be thus viewed as a nested family of its α-cuts.

A *fuzzy number* can be defined as a fuzzy set for which all α-cuts are intervals.

So, if instead of an interval $\mathbf{p}_{ij}$ of possible values of the transition probability, we have a fuzzy number $\mu_{ij}(p)$ of possible values, then we can view this information as a family of nested intervals $\mathbf{x}_{ij}(\alpha)$ (α-cuts of the given fuzzy sets).

Our objective is then to compute the fuzzy number corresponding to the desired value $p_{ij}^{(2)}$. In this case, for each level α, the corresponding α-cut of this fuzzy number can be computed based on the α-cuts $\mathbf{p}_{ij}(\alpha)$ of the corresponding input fuzzy sets. The resulting nested intervals form the desired fuzzy number for $p_{ij}^{(2)}$.

So, e.g., if we want to describe 10 different levels of uncertainty, then we must solve 10 interval computation problems. Thus, from the computational viewpoint, it is sufficient to produce an efficient algorithm for the interval case.

IV. In General, Efficient Algorithms for Computing 2-Step Transition Probabilities under Interval Uncertainty Are Known

Specifically, these algorithms are described in [11]. Motivations and details of these algorithms are given in the Appendix.

V. Symmetric Markov Chains

In some practical situations, we have symmetric (T-invariant) Markov chains, i.e., Markov chains in which the probability p_{ij} of going from state i to state j is always equal to the probability of going from state j to state i: $p_{ij} = p_{ji}$.

This happens, e.g., in describing mutations and other transitions in bioinformatics.

VI. Symmetric Markov Chains under Interval Uncertainty

As we have mentioned, in real life, we often only know the transition probabilities p_{ij} with interval (or fuzzy) uncertainty. How does the additional symmetry requirement change the range of possible values of 2-step transition probabilities?

The main change is that in the formula describing this range, we must impose this additional symmetry requirement. Thus, we arrive at the following formula:

$$\mathbf{p}^{(2)}_{ij,\rm sym} =$$

$$\left\{\sum_{k=1}^{n} p_{ik}\cdot p_{kj} : p_{ab}\in \mathbf{p}_{ab}, p_{ab}=p_{ba}, \sum_{b=1}^{n} p_{ab} = 1\right\}.$$

VII. In General, Symmetry Helps

It is known that the symmetry assumption usually enables us to speed up computations.

- First, because of symmetry, we need to store fewer data values p_{ij}.
- Second, the transition probability matrix p_{ij} become symmetric, and it is known that for symmetric matrices, there are often faster algorithms; see, e.g., [2].

It is therefore reasonable to expect that under interval uncertainty, symmetry will also be helpful.

VIII. Under Interval Uncertainty, Symmetry Makes Computations More Complex

In this paper, we prove that, contrary to the above expectations, under interval uncertainty, symmetry makes Markov chain computations more complex.

Specifically, we have mentioned that in the general case of interval uncertainty, computing 2-step transition probabilities is a computationally feasible problem. In contrast, we will prove that in the symmetric case, computing the (endpoints of the) exact range $\mathbf{p}^{(2)}_{ij,\rm sym}$ of 2-step probabilities is computationally difficult (namely, NP-hard).

IX. Proof of NP-Hardness

1°. Our proof is based on reducing, to this problem, a known NP-hard *subset* problem, where we are given n positive integers $s_1, \ldots, s_n$, and we must find the values $\varepsilon_i \in \{-1, 1\}$ for which $\sum\limits_{i=1}^{n} \varepsilon_i \cdot s_i = 0$.

For precise definitions of NP-hardness, see, e.g., [5], [18].

2°. To each instance of the subset problem, we assign a Markov chain with $[\underline{p}_{1i}, \overline{p}_{1i}] = \left[\frac{1}{n} - \alpha\cdot s_i, \frac{1}{n} + \alpha\cdot s_i\right]$ for some small $\alpha > 0$.

3°. The value α should be selected in such a way as to guarantee that the transition probabilities are always non-negative, i.e., that $\frac{1}{n} - \alpha\cdot s_i \ge 0$ for all i. This requirement is equivalent to $\alpha\cdot s_i \le \frac{1}{n}$, i.e., to $\alpha \le \frac{1}{n\cdot s_i}$. This must hold for all i, so we must make sure that α does not exceed the smallest of these values – i.e., the value corresponding to the largest s_i. Thus, we can take

$$\alpha = \frac{1}{n\cdot \max\limits_i s_i}.$$

4°. In this case, for every i, the (unknown) actual probability $p_{1i} = p_{i1}$ can be described as

$$p_{1i} = p_{i1} = \frac{1}{n} + \alpha\cdot\Delta_i,$$

where $\Delta_i \stackrel{\rm def}{=} p_{1i} - \frac{1}{n}$ can take any value from the interval $[-s_i, s_i]$.

5°. In terms of the auxiliary variables Δ_i, the requirement that $\sum\limits_{i=1}^{n} p_{1i} = 1$ means that

$$\sum_{i=1}^{n}\left(\frac{1}{n} + \alpha\cdot\Delta_i\right) = 1,$$

i.e., that $1 + \sum\limits_{i=1}^{n}\Delta_i = 1$ and $\sum\limits_{i=1}^{n}\Delta_i = 0$.

6°. Let us now find the range of possible values for the probability $p^{(2)}_{11,\rm sym}$ that in two steps we will return back from state 1 to state 1. According to the general definition of $p^{(2)}_{ij,\rm sym}$, this probability is equal to

$$p^{(2)}_{11,\rm sym} = \sum_{i=1}^{n} p_{1i}\cdot p_{i1}.$$

Since we only consider symmetric probabilities, we have $p_{i1} = p_{1i}$, hence

$$p^{(2)}_{11,\rm sym} = \sum_{i=1}^{n} p_{1i}^2.$$

Substituting the expression $p_{1i} = \frac{1}{n} + \alpha\cdot\Delta_i$ into this formula, we get

$$p^{(2)}_{11,\rm sym} = \sum_{i=1}^{n}\left(\frac{1}{n} + \alpha\cdot\Delta_i\right)^2.$$

Here,

$$\left(\frac{1}{n}+\alpha\cdot\Delta_i\right)^2=\left(\frac{1}{n}\right)^2+2\cdot\frac{1}{n}\cdot\alpha\cdot\Delta_i+\alpha^2\cdot\Delta_i^2.$$

Therefore, we conclude that

$$p^{(2)}_{11,\mathrm{sym}}=\sum_{i=1}^{n}\left(\frac{1}{n}\right)^2+\sum_{i=1}^{n}2\cdot\frac{1}{n}\cdot\alpha\cdot\Delta_i+\sum_{i=1}^{n}\alpha^2\cdot\Delta_i^2.$$

By moving constant factors outside the sum, we get:

$$p^{(2)}_{11,\mathrm{sym}}=\left(\frac{1}{n}\right)^2\sum_{i=1}^{n}1+2\cdot\frac{1}{n}\cdot\alpha\cdot\sum_{i=1}^{n}\Delta_i+\alpha^2\cdot\sum_{i=1}^{n}\Delta_i^2.$$

The first sum is equal to $\left(\frac{1}{n^2}\right)\cdot n=\frac{1}{n}$. The second sum is equal to 0 since $\sum\limits_{i=1}^{n}\Delta_i=0$. Thus, we conclude that

$$p^{(2)}_{11,\mathrm{sym}}=\frac{1}{n}+\alpha^2\cdot\sum_{i=1}^{n}\Delta_i^2.$$

7°. Let us prove that the number $\frac{1}{n}+\alpha^2\cdot\sum\limits_{i=1}^{n}s_i^2$ is a possible value of $p^{(2)}_{11,\mathrm{sym}}$ if and only if the original instance of a subset problem has a solution.

7.1°. Indeed, if the original instance has a solution ε for which $\sum\limits_{i=1}^{n}\varepsilon_i\cdot s_i=0$, then we can take $\Delta_i=\varepsilon_i\cdot s_i$ and get

$$p^{(2)}_{11,\mathrm{sym}}=\frac{1}{n}+\alpha^2\cdot\sum_{i=1}^{n}s_i^2.$$

7.2°. Vice versa, since $|\Delta_i|\le s_i$, we always have $\Delta_i^2\le s_i^2$. So, the only possibility to have

$$p^{(2)}_{11,\mathrm{sym}}=\frac{1}{n}+\alpha^2\cdot\sum_{i=1}^{n}\Delta_i^2=\frac{1}{n}+\alpha^2\cdot\sum_{i=1}^{n}s_i^2$$

is to have $\Delta_i^2=s_i^2$ for all i – otherwise, we would have

$$p^{(2)}_{11,\mathrm{sym}}=\frac{1}{n}+\alpha^2\cdot\sum_{i=1}^{n}\Delta_i^2<\frac{1}{n}+\alpha^2\cdot\sum_{i=1}^{n}s_i^2.$$

Since $\Delta_i^2=s_i^2$, we have $\Delta_i=\pm s_i$, i.e., $\Delta_i=\varepsilon_i\cdot s_i$ for some value $\varepsilon_i\in\{-1,1\}$. The fact that $\sum\limits_{i=1}^{n}\Delta_i=0$ implies that $\sum\limits_{i=1}^{n}\varepsilon_i\cdot s_i=0$. So, the values ε_i for a solution to the original instance of the subset problem.

8°. The reduction is proven, and so the problem of computing of 2-step transition probabilities in the symmetric interval-uncertainty case is indeed NP-hard.

ACKNOWLEDGMENTS

This work was supported in part by the Texas Department of Transportation grant No. 0-5453 and by the Texas Advanced Research Program Grant No. 003661-0008-2006.

The authors are thankful to the anonymous referees for valuable suggestions.

REFERENCES

[1] G. Bojadziev and M. Bojadziev, *Fuzzy sets, fuzzy logic, applications*, World Scientific, Singapore, 1995.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 2001.

[3] D. Dubois and H. Prade, "Operations on fuzzy numbers", *International Journal of Systems Science*, 1978, Vol. 9, pp. 613–626.

[4] W. J. Ewens and G. R. Grant, *Statistical Methods in Bioinformatics: An Introduction*, Springer-Verlag, 2005.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, California, 1979.

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

[7] 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.

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

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

[10] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic*, Prentice Hall, New Jersey, 1995.

[11] O. Kosheleva, M. Shpak, M. A. Campos, G. P. Dimuro, and A. C. da Rocha Costa, "Computing Linear and Nonlinear Normal Modes under Interval (and Fuzzy) Uncertainty", *Proceedings of the 25th International Conference of the North American Fuzzy Information Processing Society NAFIPS'2006*, Montreal, Quebec, Canada, June 3–6, 2006.

[12] T. Koski, *Hidden Markov Models for Bioinformatics*, Springer-Verlag, 2001.

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

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

[15] R. E. Moore and W. A. Lodwick, "Interval analysis and fuzzy set theory", *Fuzzy Sets and Systems*, 2003, Vol. 135, No. 1, pp. 5–9.

[16] H. T. Nguyen and V. Kreinovich, "Nested intervals and sets: concepts, relations to fuzzy sets, and applications", In: R. B. Kearfott and V. Kreinovich (eds), *Applications of interval computations*, Kluwer, Dordrecht, pp. 245–290.

[17] H. T. Nguyen and E. A. Walker, *A First Course in Fuzzy Logic*, CRC Press, Boca Raton, Florida, 2006.

[18] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, Massachusetts, 1994.

[19] D. Škulj, "Finite discrete time Markov chains with interval probabilities", In: In: J. Lawry, E. Miranda, A. Bugarin, S. Li, M. A. Gil, P. Grzegorzewski, and O. Hryniewicz (eds.), *Soft Methods for Integrated Uncertainty Modeling*, Springer-Verlag, 2006, pp. 299–306.

APPENDIX: ANALYSIS OF THE GENERAL (NON-SYMMETRIC) PROBLEM AND A STEP-BY-STEP DESIGN OF EFFICIENT ALGORITHMS FOR SOLVING THIS PROBLEM

Reduction to SUE expressions. We would like to use interval computations (see, e.g., [7], [8], [9], [14]) in our estimates. 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., [6]). To avoid

this excess width, let us first represent the expression (2) 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:

$$p_{ii}^{(2)} = \sum_{k \neq i} p_{ik} \cdot p_{kj} + p_{ii}^2. \tag{3}$$

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

$$p_{ij}^{(2)} = \sum_{k \neq i,j} p_{ik} \cdot p_{kj} + p_{ij} \cdot (p_{ii} + p_{jj}). \tag{4}$$

Auxiliary peeling algorithm for solving quadratic optimization problems: reminder. To compute the exact range of $p_{ij}^{(2)}$, we must find the maximum and the minimum of the corresponding expressions (3) and (4) under the conditions (1) and

$$\underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij}. \tag{5}$$

In other words, we want to optimize a quadratic function under linear constraints (equalities and inequalities).

To solve these auxiliary optimization problems, we can use the idea of peeling; see, e.g, [8]. The idea of peeling is a natural extension of the known simplex techniques for solving linear programming problems. This idea can be described as follows.

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 $\approx 2^m$ of them, where m is the overall number of variables and constraints. 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:

$$\frac{\partial E}{\partial x_{i_k}} = 0. \tag{6}$$

Since the derivative of a quadratic function is a linear function, the equations (6) 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.)

The only problem is that since we have $\approx 2^n$ boundary elements, this algorithm requires exponential ($\approx 2^n$) time. So, we must design a faster 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 (3). In this case, we want to solve the following problem:

$$\sum_{k \neq i} p_{ik} \cdot p_{ki} + p_{ii}^2 \to \max \tag{7}$$

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

$$\sum_{k \neq i} p_{ik} + p_{ii} = 1. \tag{8}$$

In (7), 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 (7) is also a solution to the following problem with fewer unknowns:

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

under the conditions (8) and

$$\underline{p}_{ik} \leq p_{ik} \leq \overline{p}_{ik}. \tag{10}$$

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:

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

under the conditions (10) and

$$\sum_{k \neq i} p_{ik} = 1 - p_{ii}. \tag{12}$$

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

$$\overline{p}_{(1)i} \geq \overline{p}_{(2)i} \geq \ldots \geq \overline{p}_{(n-1)i}. \tag{13}$$

The sums in (11) and (12) do not depend on the order in which we add the terms. Thus, the above optimization problem can be reformulated as follows:

$$\sum_{k} p_{i(k)} \cdot \overline{p}_{(k)i} \to \max \tag{14}$$

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

$$\underline{p}_{i(k)} \le p_{i(k)} \le \overline{p}_{i(k)} \tag{15}$$

and

$$\sum_k p_{i(k)} = 1 - p_{ii}. \tag{16}$$

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 (16) 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 (14) 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:

$$p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ii}^2 \to \max \tag{17}$$

under the conditions that

$$\underline{p}_{i(k')} \le p_{i(k')} \le \overline{p}_{i(k')}, \quad \underline{p}_{ii} \le p_{ii} \le \overline{p}_{ii}, \tag{18}$$

and

$$p_{i(k')} + p_{ii} = c_{k'}, \tag{19}$$

where we denoted

$$c_{k'} \stackrel{\text{def}}{=} 1 - \sum_{k:k<k'} \overline{p}_{i(k)} - \sum_{k:k>k'} \underline{p}_{i(k)}. \tag{20}$$

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

$$V_{k'} = \sum_{k:k<k'} \overline{p}_{i(k)} \cdot \overline{p}_{(k)i} + p_{i(k')} \cdot \overline{p}_{(k')i} + \sum_{k:k>k'} \underline{p}_{i(k)} \cdot \overline{p}_{(k)i} + p_{ii}^2. \tag{21}$$

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 $\overline{p}_{ki}$ ($k \ne i$) in decreasing order: $\overline{p}_{(1)i} \ge \overline{p}_{(2)i} \ge \ldots \ge \overline{p}_{(n-1)i}$.
- Then, for each $k' = 1, \ldots, n-1$, we do the following:
  - we compute the value $c_{k'}$ by using formula (20);
  - we solve the problem (17)–(19) 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 (21).
- 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., [2]). 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 (2). 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

$$W_{k'} \stackrel{\text{def}}{=} \sum_{k:k<k'} \overline{p}_{i(k)} \cdot \overline{p}_{(k)i} + \sum_{k:k>k'} \underline{p}_{i(k)} \cdot \overline{p}_{(k)i}, \tag{22}$$

for which

$$V_{k'} = W_{k'} + p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ii}^2. \tag{23}$$

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

$$c_1 = 1 - \sum_{k:k>1} \underline{p}_{i(k)} \tag{24}$$

and

$$W_1 = \sum_{k:k>1} \underline{p}_{i(k)} \cdot \overline{p}_{(k)i}. \tag{25}$$

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:

$$c_k = c_{k-1} - \overline{p}_{i(k-1)} + \underline{p}_{i(k)} \tag{26}$$

and

$$W_k = W_{k-1} + \overline{p}_{i(k-1)} \cdot \overline{p}_{(k-1)i} - \underline{p}_{i(k)} \cdot \overline{p}_{(k)i}. \tag{27}$$

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 (24); for $k' > 1$, we use the formula (26);
- we solve the problem (17)–(19) 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 (25); for $k' > 1$, we use the formula (27);
- based on the solution of the quadratic optimization problem, we compute $V_{k'}$ by using the formula (23).

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 (4), lead to the conclusion that the desired upper endpoints is a solution to the following simplified optimization problem:

$$\sum_{k \neq i,j} p_{ik} \cdot \overline{p}_{kj} + p_{ij} \cdot (p_{ii} + \overline{p}_{jj}) \to \max \tag{28}$$

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

$$\sum_{k \neq i,j} p_{ik} + p_{ii} + p_{ij} = 1. \tag{29}$$

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:

$$p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ij} \cdot (p_{ii} + \overline{p}_{jj}) \to \max \tag{30}$$

under the conditions that

$$\underline{p}_{i(k')} \leq p_{i(k')} \leq \overline{p}_{i(k')}, \quad \underline{p}_{ii} \leq p_{ii} \leq \overline{p}_{ii}, \tag{31}$$

$$\underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij}, \quad p_{i(k')} + p_{ii} + p_{ij} = c_{k'}, \tag{32}$$

where we denoted

$$c_{k'} \stackrel{\text{def}}{=} 1 - \sum_{k:k<k'} \overline{p}_{i(k)} - \sum_{k:k>k'} \underline{p}_{i(k)}. \tag{33}$$

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

$$V_{k'} = W_{k'} + p_{i(k')} \cdot \overline{p}_{(k')i} + p_{ij} \cdot (p_{ii} + \overline{p}_{jj}). \tag{34}$$

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 (24); for $k' > 1$, we use the formula (26);
- we solve the problem (30)–(**??**) 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 (25); for $k' > 1$, we use the formula (27);
- based on the solution of the quadratic optimization problem, we compute $V_{k'}$ by using the formula (34).

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

$$n \cdot O(n \cdot \log(n)) = O(n^2 \cdot \log(n)) \ll O(n^3) \tag{35}$$

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.

As a result, we arrive at the $O(n^3)$ algorithms for computing 2-step transition probabilities for interval Markov chains that are described in [11].