

Sorting processes with energy-constrained comparisons*

Barbara Geissmann and Paolo Penna

Department of Computer Science, ETH Zurich, Zurich, Switzerland



(Received 6 September 2017; revised manuscript received 23 February 2018; published 8 May 2018)

We study very simple sorting algorithms based on a *probabilistic* comparator model. In this model, errors in comparing two elements are due to (1) the energy or effort put in the comparison and (2) the difference between the compared elements. Such algorithms repeatedly compare and swap pairs of randomly chosen elements, and they correspond to natural Markovian processes. The study of these Markov chains reveals an interesting phenomenon. Namely, in several cases, the algorithm that repeatedly compares only *adjacent* elements is *better* than the one making arbitrary comparisons: in the long-run, the former algorithm produces sequences that are “better sorted”. The analysis of the underlying Markov chain poses interesting questions as the latter algorithm yields a nonreversible chain, and therefore its stationary distribution seems difficult to calculate explicitly. We nevertheless provide bounds on the *stationary distributions* and on the *mixing time* of these processes in several restrictions.

DOI: [10.1103/PhysRevE.97.052108](https://doi.org/10.1103/PhysRevE.97.052108)

I. INTRODUCTION

Statistical mechanics deals with complex systems made of a large number of simple interacting particles whose behavior is inherently probabilistic. In such systems, the *temperature* governs the probabilistic behavior of the simple interactions, and ultimately reflects into a global state of the system when it reaches an *equilibrium*. Typically, low temperature results in ordered configurations, and high temperature results in disordered ones.

In this work, we use a similar approach to model a basic problem in computation, namely, the problem of *sorting* elements using *erroneous* pairwise comparisons (see below). Roughly speaking, in our model the “temperature” corresponds to the “error probability” of a single comparison, and each comparison corresponds to an “interaction” between two different particles—numbers—which swap their position according to the result of the comparison:

higher temperature \leftrightarrow higher error probability,
particle interaction \leftrightarrow pairwise comparison.

Suppose one has to sort a number of elements by making pairwise comparisons, but sometimes the result of a comparison is *incorrect*. Errors are often unavoidable, or even deliberately introduced to save other important resources. For example, errors occur in measurements that require high precision (where a small noise can affect the result) or judgment made by individuals (who naturally tend to make small mistakes) [2–4]. Also, when constructing integrated circuits with *probabilistic complementary metal-oxide-semiconductors* (abbreviated as PCMOs), it is possible to trade *energy* for *errors*, that is, one can reduce the energy spent for a single operation but this will increase the probability of incorrect response [5]. One can thus envision the following situation:

(1) It is “easier” to compare two elements if they differ “a lot”, while errors are more likely when they are “very close.”

(2) If we are able or willing to spend more energy (effort) on a single comparison, then we can increase the probability of getting the correct result.

Given this scenario, one would like to decide what strategies are good to sort the elements nearly correctly. In particular, we ask the following question:

How is the performance/result influenced by different strategies (algorithms)?

The problem of sorting has been addressed under various models of errors (see, e.g., Refs. [6–8]). The purpose of this work is to study the above question under an error model which captures the two features described in Items 1 and 2. To this end, we consider a very simple type of algorithms based on repeated comparisons and swaps of randomly chosen elements.

A. Our contribution

In this work, we look at extremely simple *sorting* processes on what we call a *probabilistic comparator* model, which is inspired by classical models in *statistical physics* (see below for details on our model). When sorting a set of elements, one performs several comparisons between pairs of elements for a certain number of steps. At each step, the compared elements are swapped or not according to the result of their comparison. Comparing and swapping *identical* elements does *not change* the sequence. Consider the following two simple algorithms:

(1) *Arbitrary swaps*. Compare two randomly chosen elements of the sequence.

(2) *Adjacent swaps*. Compare a randomly chosen element with the next one in the sequence.

We experimentally observed the following interesting phenomenon (see Fig. 1):

*A technical report version of this work appeared in Ref. [1].

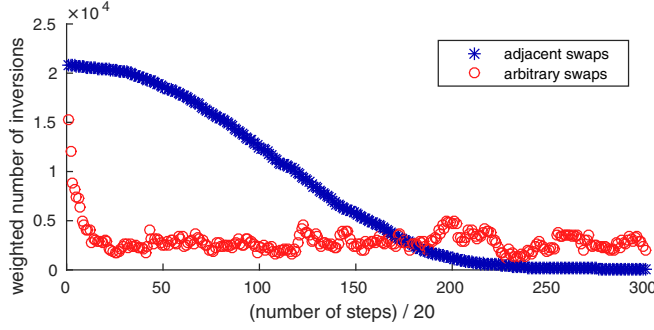


FIG. 1. Comparison of two simple algorithms for sorting: on the long run the algorithm doing only adjacent comparisons gives a better result compared to the one doing all comparisons. The weighted number of inversions (y axis) is a measure of the “disorder” of the current sequence. Both algorithms operate under the same conditions (same input sequence and identical error probabilities of comparisons). In particular, the input sequence is $(50, 49, \dots, 1)$ which has the highest weighted number of inversions, while the sorted sequence is $(1, 2, \dots, 50)$ with zero weighted number of inversions. The probability of errors is controlled by a parameter λ (in this experiment $\lambda = e^{1/15}$, though the same behavior occurs for essentially any fixed λ as shown in Sec. VI).

The algorithm performing only *adjacent swaps* gives better results than the one with *arbitrary swaps* after a certain number of comparisons are made.

Intuitively, the y axis in Fig. 1 measures the “disorder” in the current sequence in each algorithm. Note that the initial input sequence is irrelevant as long as we consider sufficiently many steps of such algorithms. This is perhaps the second important observation that one can make looking at Fig. 1:

The *time* for these two processes to reach an equilibrium seems also significantly different, as the “less accurate” *arbitrary swaps* algorithm is *faster*.

These properties can be formally captured by viewing these algorithms as *Markov chains*, and by analyzing their *stationary distribution* and their *mixing time*. As we discuss below, the analysis of the chain with adjacent swaps leads to an interesting notion of “disorder” of a sequence which we call *weighted number of inversions*. We next describe more in detail our contributions.

Probabilistic comparator model. We introduce a simple model in which the probability that a comparison between two elements (numbers) is correct depends on two factors: (1) how different the two elements are and (2) the “effort” or “energy” spent to make the comparison. Intuitively speaking, the more energy the more accurate is the comparison, meaning that even very similar elements (small difference) can be distinguished. More precisely, for an energy parameter $\lambda \geq 1$, a comparison between two numbers a and b is correct with probability

$$p_{ab} := \frac{\lambda^{b-a}}{\lambda^{a-b} + \lambda^{b-a}},$$

where b is the biggest between a and b . One should think about a single comparison as a measurement of two quantities, which sometimes can be erroneous especially when the difference is small. All comparisons (including those involving the same

two numbers) are independent and performed with the same parameter λ . That is, we consider *independent* errors in which the error probabilities are described by the formula above.

Remark 1 (Our model, statistical physics, and further connections). Our comparison model is inspired by classical models in *statistical physics* [9] and in *game theory* [3], where $\lambda = e^{1/\text{noise}}$. In statistical physics, the noise parameter is the “temperature” of the system, and high temperature corresponds to highly disordered configurations. In game theory, $1/\text{noise}$ represents to the “rationality level” of the players, i.e., their ability to distinguish between strategies with similar payoffs. To some extent, this model can be seen as an abstraction of the *probabilistic CMOS* technology which allows to trade energy for correctness [5]. As our model attempts to abstract from hardware-specific construction details, it necessarily introduces certain simplifying assumptions. Arguably, the major of these assumptions is that the probability of errors depends uniquely on the difference between the two numbers to be compared, and not on their actual values. However, the parameter λ captures the property that, by increasing the energy per single operation, the probability of correct comparisons increases in some way (depending on the hardware). A similar model in which “close” elements are difficult to distinguish is the one in Ref. [2], where comparisons are *always correct* if the difference between the compared elements is above a *certain threshold* (see also Ref. [4]).

Algorithms and Markov chains. The two algorithms above (and others) can be viewed as Markov chains whose stationary distribution describes the output if we let them run long enough. We believe that these processes are interesting by themselves (for instance, they can be seen as variants of other well-studied processes—see Sec. IB), and they pose new questions on how to analyze them. We first show that the adjacent swaps algorithm corresponds to a *reversible* Markov chain \mathcal{M}_{adj} and its stationary distribution has a simple closed formula of the form

$$\pi(s) \propto \lambda^{-2w(s)}, \quad (1)$$

where $w()$ is what we call the *weighted number of inversions* of sequence s ,

$$w(s) := \sum_{i < j: s_i > s_j} s_i - s_j,$$

a measure of its “distance” from the correctly sorted sequence. Intuitively, inversions involving very different elements count more than inversions of almost identical elements. In contrast, the algorithm with arbitrary (random) pair comparisons corresponds to a *nonreversible* chain \mathcal{M}_{any} , and therefore the analysis of its stationary distribution is considerably more complicated.

Figure 1, and all experiments we made on various input sequences and parameters (see the Appendix), suggest that \mathcal{M}_{adj} yields *better sorted* sequences than \mathcal{M}_{any} , though the latter chain *converges faster* to its stationary distribution. We study both the *mixing time* and the properties of the *stationary distribution*, like the probability of returning to the sorted sequence. Since \mathcal{M}_{any} is nonreversible and difficult to analyze, we focus on special cases of sequences where we

TABLE I. Overview of our results (from “simpler” to “harder” instances): The table contains rows, each for a different type of input instances: *One outlier* sequences contain $n - 1$ a ’s and 1 $b > a$. *Binary Input* sequences contain n_a a ’s and n_b b ’s, such that $n = n_a + n_b$. *Three elements* sequences contain three elements, such that not all three of them are equal. In column two and three we compare the two Markovian processes \mathcal{M}_{adj} and \mathcal{M}_{any} , respectively, on the different types of inputs: Our results include bounds on the *mixing time* of the Markov chains (for *One outlier* and *Binary inputs*), the *probability to sort correctly* (for *One outlier* and *Three elements*), and the *expected weighted number of inversions* in the stationary distribution (for *One outlier*). For every result, the corresponding theorem is stated in the third column.

	Adjacent swaps (\mathcal{M}_{adj})	Arbitrary swaps (\mathcal{M}_{any})	
One outlier (containing $n - 1$ a ’s and 1 $b > a$, error probability $p = p_{ba}$)	Mixing time		
	$O(n^2)$	$O(n)$	Theorems 12 and 15
	Probability to sort correctly		
	Constant ($> \frac{1-2p}{1-p}$)	Vanishing ($< \frac{1-p}{np}$)	Theorem 16
	Expected weighted number of inversions		
	$O(1)$	$\Omega(n)$	Theorem 16
Binary inputs (containing n_a a ’s and n_b b ’s)	Mixing time		
	$O(n^2)$	$O(n \log n')$ ($n' = \min\{n_a, n_b\}$)	Theorems 12 and 15
Three elements (not all of them are equal)	Probability to sort correctly		
	Better than \mathcal{M}_{any}	Worse than \mathcal{M}_{adj}	Theorem 20

either reduce their size or number of distinct elements (see Table I for an overview of our results):

(1) *Mixing time*. In Sec. IV, we consider the case of *binary sequences*, where each element in the sequence is either a or b . We show that the mixing time of \mathcal{M}_{adj} is $O(n^2)$, while for \mathcal{M}_{any} it is $O(n \log n)$, or even linear if the number of occurrences of b is constant, for every $\lambda > 1$.

(2) *Sortedness*. In Sec. V, we study the probability that the chains return the sorted sequence at stationary distribution. We show that \mathcal{M}_{adj} is better than \mathcal{M}_{any} when sorting *three arbitrary elements*. This result is based on the *Markov chain tree theorem* and it is the most involved in this section. Similar results hold also for sorting arbitrary long sequences with a *single outlier*, that is, binary sequences with a single element $b > a$ and many a ’s. Here the analysis shows a quantitative difference between the two chains (cf. Theorem 16).

We also provide a variant of \mathcal{M}_{any} in which comparisons of nonadjacent pairs are done *multiple times*: For example, if numbers a and b are two positions away in the current sequence (say $a = s_i$ and $b = s_{i+2}$) then we perform *two* comparisons and accept to swap them only if *both* of them tell to do so. We show that this third chain $\mathcal{M}_{\text{any}}^*$ has the *same stationary distribution* as \mathcal{M}_{adj} (Theorem 10). This implies that all the results on sortedness apply also to $\mathcal{M}_{\text{any}}^*$ in place of \mathcal{M}_{adj} . Therefore, one can see this “careful swapping” rule as a way to fix the algorithm doing naively arbitrary swaps.

Remark 2 (Related processes in statistical physics). The processes considered in this work, besides being quite natural, generalize the well-studied *asymmetric simple exclusion process*—ASEP—see, e.g., Refs. [10–16]. Specifically, what we call in this paper the *binary case*, i.e., sorting sequences made of only two different types of elements, a and b , performing only *adjacent* comparisons, is precisely the ASEP. This is the simplest of our processes, and generalizations are obtained when performing nonadjacent comparisons and/or considering sequences in which all elements are different from each other.

The ASEP is a special case of models known as driven lattice gases [17,18], which describe systems of hopping and interacting particles, i.e., particles moving at random to neighboring positions if that position is unoccupied. The ASEP has many applications in physical chemistry and statistical mechanics, including protein synthesis [12], traffic flow problems [19], growth models [20], and vertex models [21,22].

B. Related work

Stochastic models of the form of Eq. (1) are very common in statistics and, in particular, Mallows [23] was among the firsts to consider such models in the context of permutations: Their weight function $w()$ is a suitable distance function which comes from probabilities p_{ab} of ranking a before b . In that sense, our model is a special case of Mallows’, though the procedure of Ref. [23] is different: all pairwise comparisons are made at once until a consistent result is obtained. Our probabilistic comparator is also a special case of Bradley and Terry’s [24], where the probability p_{ab} is of the form $\frac{w_a}{w_a + w_b}$ (our model corresponds to $w_a = \lambda^{2a}$).

Several restrictions on p_{ab} have been studied for the natural Markov chain which makes only adjacent comparisons. The classical card shuffling problem corresponds to the *unbiased* version of this chain in which all probabilities p_{ab} equal $1/2$, for which Wilson [25] proved that this chain is rapidly mixing and gave a very tight bound. A similar problem is the uniform sampling of partial order extensions, which corresponds to probabilities p_{ab} being $1/2$ or 1 and $p_{ba} = 1 - p_{ab}$. For the latter, Bubley and Dyer [26] showed that this chain is also rapidly mixing. Benjamini *et al.* [27] proved rapidly mixing for the *constant biased* case, that is, when every comparison is correct with some fixed probability $p > 1/2$, independently of the compared elements: $p_{ab} = p > 1/2$ for all $a < b$. The mixing time of *biased* comparisons has been studied by Bhakta *et al.* [28] under two comparison models called “choose your weapon” and “league hierarchies”: In the first model, p_{ab}

depends only on the largest between a and b , while in the second model all numbers are the leaves of some tree and p_{ab} depends only on the least common ancestor of a and b . Note that our model does not fall in either class even for the case of only three distinct elements. A recent work by Haddadan and Winkler [29] studied the process with adjacent comparisons on *ternary inputs* (linear particle systems with three particles types, in their terminology) and showed that the mixing time of this Markov chain is at most $O(n^{10})$. This result is used to show that the process which distinguishes swaps between identical elements (called *gladiator chain* in that work) is also rapidly mixing, where the upper bound on the mixing time is $O(n^{18})$.

The phenomenon observed in our work (Fig. 1), has been subsequently studied by Gavenčiak *et al.* [30] for the *constant biased* case (constant probability $p > 1/2$ of correct comparison). They indeed proved that adjacent swaps result in sequences with fewer inversions than any pairs swaps. The weighted number of inversions, which arises in our work from the analysis of the adjacent swaps chain, turns out to be a useful for Ref. [30] in bounding the mixing time of their process.

Diaconis and Ram [31] studied a different type of chains called systematic scan algorithms: for the *unbiased* case, they proved that n of such scan operations are sufficient to reach the stationary distribution.

C. Preliminary definitions on Markov chains

In this section, we introduce some of the definitions on Markov chains used throughout this work (for more details, see Ref. [32]). A Markov chain over a finite state space S is specified by a transition matrix P , where $P(s, s')$ is the probability of moving from state s to state s' in one step. The t th power of the transition matrix gives the probability of moving from one state to another state in t steps. All chains studied in this work are ergodic meaning that they have a unique *stationary distribution* π : for any two states s and s' , $\lim_{t \rightarrow \infty} P^t(s, s') = \pi(s')$. The *mixing time* of a Markov chain is the time needed for $P^t(s, \cdot)$ to get “sufficiently close” to π for any starting state s :

$$t_{\text{mix}}(\epsilon) := \min_{t \in \mathbb{N}} \max_{s \in S} \{ \| P^t(s, \cdot) - \pi \|_{TV} \leq \epsilon \},$$

where $\| P^t(s, \cdot) - \pi \|_{TV} = \frac{1}{2} \sum_{s' \in S} |P^t(s, s') - \pi(s')|$ is the total variation distance.

We will use the definition of a *reversible* Markov chain, also called *detailed balanced condition*: If the transition matrix P admits a vector π such that $\pi(s)P(s, s') = \pi(s')P(s', s)$ for all s and s' , then π is the *stationary distribution* of the chain with transitions P .

An equivalent characterization of reversible chains is given by looking at cycles over the states. For any subset $\Gamma \subseteq S \times S$ of transitions (pairs of states of the chain), define the associated probability as the product of all these transitions in the chain,

$$\mathbf{P}(\Gamma) := \prod_{(x, y) \in \Gamma} P(x, y). \quad (2)$$

Let Γ^{-1} denote the reversed edges in Γ , that is, $\Gamma^{-1} := \{(y, x) | (x, y) \in \Gamma\}$. The *Kolmogorov reversibility criterion* [33] says that a chain is reversible if and only if for any

cycle \mathcal{C} ,

$$\mathbf{P}(\mathcal{C}) = \mathbf{P}(\mathcal{C}^{-1}). \quad (3)$$

For the sake of clarity, we sometimes denote cycles as $\mathcal{C} = s^1 \rightarrow s^2 \rightarrow \dots \rightarrow s^\ell \rightarrow s^1$ and the corresponding reversal by $\mathcal{C}^{-1} = s^1 \leftarrow s^\ell \leftarrow \dots \leftarrow s^2 \leftarrow s^1$.

The stationary distribution of any (even nonreversible) Markov chain can be computed by looking at the probabilities of all directed trees rooted at some state. More formally, let $\mathcal{T}(s)$ be the set of all directed trees rooted at state s , that is, from every other state there is a path towards s in the tree. The *Markov chain tree theorem* (see Ref. [34], Chapter 6, Lemma 3.1) says that, for any ergodic Markov chain with transition matrix P , its stationary distribution π is given by

$$\pi(s) = \frac{W(s)}{\sum_{\hat{s} \in S} W(\hat{s})}, \quad \text{where} \quad W(s) := \sum_{T \in \mathcal{T}(s)} \mathbf{P}(T). \quad (4)$$

II. A MEASURE OF DISORDER (WEIGHTED NUMBER OF INVERSIONS)

It turns out that the analysis of the stationary distribution of the algorithm performing only adjacent swaps leads to an interesting measure of disorder. In this section, we introduce a formal definition of such a measure, which we call the *weighted number of inversions*. The next section will connect this notion to the stationary distribution of the algorithm mentioned above.

Definition 3. The weighted number of inversions of a sequence s is defined as

$$w(s) := \sum_{i < j: s_i > s_j} s_i - s_j.$$

Example 4. Consider the sequence $s = (5, 2, 3)$ and the sorted sequence $(2, 3, 5)$. Then the weighted number of inversions of s is equal to $w(s) = (5 - 2) + (5 - 3) = 5$.

The displacements of the single elements allow an equivalent way to describe the weighted number of inversions (this equivalent definition turns out to be useful in the next section).

Lemma 5. For a sequence s , let $s^{(\text{sort})}$ be the sequence sorted in nondecreasing order. Then, $w(s) = \sum_i (s_i^{(\text{sort})} - s_i) i$.

Proof. In the sum $\sum_{i < j: s_i > s_j} s_i - s_j$, every element s_i is added r_i and subtracted l_i times, where r_i is the number of smaller elements on its right-hand side and l_i the number of larger elements on its left. Thus,

$$\sum_{i < j: s_i > s_j} s_i - s_j = \sum_i (r_i - l_i) s_i.$$

The difference $d_i = r_i - l_i$ corresponds to the displacement of s_i to the left compared to the sorted sequence; i.e., $s_{i+d_i}^{(\text{sort})} = s_i$. Note that with this notation we have that $w(s) = \sum_i d_i s_i$.

Consider now the sum $\sum_i (s_i^{(\text{sort})} - s_i) i = (\sum_i s_i^{(\text{sort})} i) - (\sum_i s_i i)$. Each element $s_i = s_{i+d_i}^{(\text{sort})}$ appears twice in this quantity: once multiplied by $i + d_i$ in the first summation and once multiplied by $-i$ for the second one. The contribution of each s_i to the quantity is thus exactly $d_i s_i$, and therefore $\sum_i (s_i^{(\text{sort})} - s_i) i = \sum_i d_i s_i = w(s)$. ■

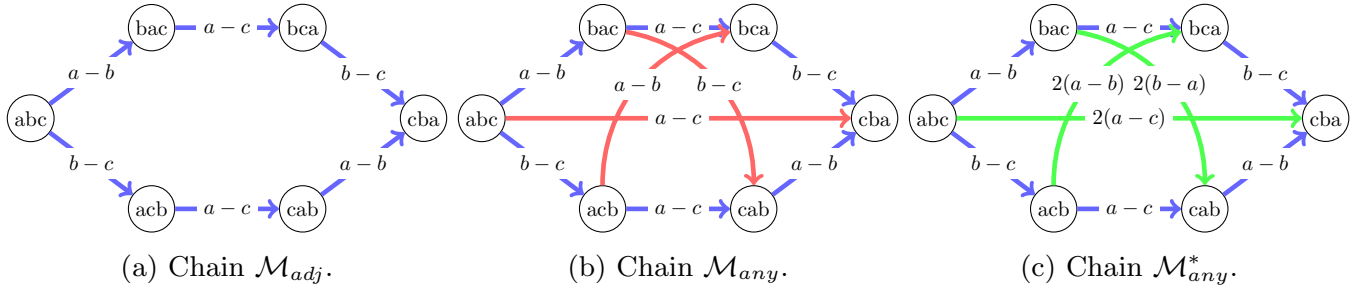


FIG. 2. The three chains for sorting three elements abc ; A transition with label w has probability $\frac{\lambda^w}{\lambda^w + \lambda^{-w}}$; for clarity, we only show forward transitions.

III. SORTING ALGORITHMS AS MARKOV CHAINS

In this section we define the algorithms and the resulting Markov chains. The first chain performs only adjacent comparisons, whereas the second and the third chains allow us to compare and swap any two numbers of the sequence. In Fig. 2, each chain is illustrated by a three-elements example.

Definition 6. The chain \mathcal{M}_{adj} (adjacent swaps) is defined as follows:

- (1) Pick an index i in $\{1, \dots, n-1\}$ uniformly at random;
- (2) Swap $s_i = a$ and $s_{i+1} = b$ with probability $\frac{\lambda^{a-b}}{\lambda^{a-b} + \lambda^{b-a}}$.

Definition 7. The chain \mathcal{M}_{any} (arbitrary swaps) is defined as follows:

- (1) Pick two indexes i and j in $\{1, \dots, n\}$ uniformly at random, with $i < j$;

- (2) Swap $s_i = a$ and $s_j = b$ with probability $\frac{\lambda^{a-b}}{\lambda^{a-b} + \lambda^{b-a}}$.

Definition 8. The chain $\mathcal{M}_{\text{any}}^*$ (arbitrary swaps reversible) is defined as \mathcal{M}_{any} except that the probability of swapping $s_i = a$ and $s_j = b$ is $(\frac{\lambda^{a-b}}{\lambda^{a-b} + \lambda^{b-a}})^{j-i}$.

The above processes are quite natural, and they can also be seen in a slightly different way. In particular, the chain \mathcal{M}_{adj} is equivalent to both the following two processes:

- (1) Pick a pair of consecutive positions i and $i+1$ uniformly at random [probability $1/(n-1)$] and compare the elements in position i and $i+1$.
- (2) Pick an index i uniformly at random in $\{2, \dots, n\}$ and always compare the element in this position with the *previous* one (index $i-1$), instead of the next one as in Definition 6.

Another variant would be to allow both *next* and *previous* comparisons, that is, to pick a random element and compare it with one of its two neighbors chosen at random:

- (1) Pick an index i uniformly at random in $\{1, 2, \dots, n\}$, and a *random direction* $r \in \{-1, +1\}$ (previous or next). Compare the element in position i with that in position $i+r$ if the latter is a valid position, i.e., $i+r \in \{1, \dots, n\}$; otherwise, whenever $i+r$ is not a valid position, do nothing.

The above process can be intuitively thought as one “random particle” interacting with one of its two “neighbors” chosen also at random. The “interaction” is the comparison whose result decides if the two particles swap their relative positions. This process is simply a *lazy* version of \mathcal{M}_{adj} since it chooses which comparison to perform as follows:¹

¹To see that this is indeed the case, note that each of the $n-1$ pairs $(i, i+1)$ is chosen with probability $1/n$, and with probability $1 - 1/n$ we do nothing.

- (1) With probability $1/n$ do nothing;
- (2) With probability $1 - 1/n$ pick one of the $n-1$ possible pairs $(i, i+1)$ with uniform distribution (thus make one step of chain \mathcal{M}_{adj}).

The stationary distribution of \mathcal{M}_{adj} automatically gives us the stationary distribution of the process above as they are the same. Analogously, for \mathcal{M}_{any} and $\mathcal{M}_{\text{any}}^*$ one can define a lazy variant in which, after picking a random index i , we choose a random direction and distance for the comparison: pick $r \in \{-n+1, n-1\}$ uniformly at random and compare elements in position i and $i+r$ if the latter is a valid position (and do nothing otherwise). In this case, each pair (i, j) with $i < j$ is chosen with probability $\frac{1}{n} \frac{1}{2(n-1)} + \frac{1}{n} \frac{1}{2(n-1)} = \frac{1}{n(n-1)}$, and with probability $\frac{1}{n}$ we do nothing. The probabilities of actually swapping the compared elements are the same as in definitions of \mathcal{M}_{any} and $\mathcal{M}_{\text{any}}^*$, respectively.

In both cases, the lazy version of the chains is just a constant factor slower than its original counterpart, therefore asymptotic bounds in the mixing time of the original chains apply also to the lazy variant. Furthermore, the processes are indeed equivalent in the sense that the chains converge to the same stationary distribution (see, e.g., Ref. [32]).

A. (Non-)Reversibility and stationary distribution

We show below that both \mathcal{M}_{adj} and $\mathcal{M}_{\text{any}}^*$ are reversible and possess the same stationary distribution. Moreover, we prove that this stationary distribution assigns higher probabilities to the sequences that are “nearly sorted” as it corresponds exactly to the weighted number of inversions (see Sec. II). Finally, we show that the chain \mathcal{M}_{any} is instead nonreversible.

Theorem 9. The chain \mathcal{M}_{adj} is reversible with stationary distribution $\pi(s) \propto \lambda^{-2w(s)}$, where $w(s)$ is the weighted number of inversions.

Proof. We prove that \mathcal{M}_{adj} is reversible. Let s and s' be two sequences that differ in i ’s swap [otherwise, $P(s, s') = 0 = P(s', s)$ and reversibility is trivial]. Observe that by definition

$$\frac{P(s, s')}{P(s', s)} = \lambda^{2(s_i - s_{i+1})} = \lambda^{2(a-b)} \quad \text{and} \quad \frac{\pi(s')}{\pi(s)} = \lambda^{2w(s) - 2w(s')}.$$

Since s' is obtained from s by swapping $a = s_i$ and $b = s_{i+1}$,

$$\begin{aligned} w(s) - w(s') &= (s_i^{(\text{sort})} - s_i)i - (s_i^{(\text{sort})} - s'_i)i \\ &\quad + (s_{i+1}^{(\text{sort})} - s_{i+1})(i+1) \\ &\quad - (s_{i+1}^{(\text{sort})} - s'_{i+1})(i+1) \\ &= a - b, \end{aligned}$$



FIG. 3. Idea of the coupling (proof of Lemma 14). On the left: arrows show how to obtain sequences $x_{j^* \times k}$ and $y_{j^* \times k}$, respectively. On the right: arrows show how to obtain sequences $x_{j^* \times k}$ and $y_{j^* \times k}$, respectively. In either case, the resulting sequences are identical.

where $s^{(\text{sort})}$ is the sorted sequence. Therefore the detailed balance condition is satisfied. ■

Theorem 10. The chain $\mathcal{M}_{\text{any}}^*$ is reversible and has the same stationary distribution as \mathcal{M}_{adj} .

Proof. To prove that $\mathcal{M}_{\text{any}}^*$ has the same stationary distribution as \mathcal{M}_{adj} , consider any transition from s to s' which swaps two elements at distance $k \geq 1$. There exists a path \mathcal{P} in \mathcal{M}_{adj} that leads from s to s' and whose probability equals the one of the single swap in $\mathcal{M}_{\text{any}}^*$,

$$\mathbf{P}(\mathcal{P}) = \prod_{(x,y) \in \mathcal{P}} P(x,y) = \frac{\lambda^{k(a-b)}}{D(\mathcal{P})}.$$

The path is obtained by simulating the swap between a and b via adjacent swaps:

$$\begin{aligned} \mathcal{P} &:= (\dots ax_1 \dots x_{k-1} b \dots) \rightarrow (\dots x_1 a \dots x_{k-1} b \dots) \dots \\ &\rightarrow (\dots x_1 \dots x_{k-1} ab \dots) \rightarrow (\dots x_1 \dots x_{k-1} ba \dots) \\ &\rightarrow (\dots x_1 x_2 \dots bx_{k-1} a \dots) \dots \\ &\rightarrow (\dots bx_1 x_2 \dots x_{k-1} a \dots), \end{aligned}$$

which yields in the numerator the product

$$\lambda^{(a-x_1)+\dots+(a-x_{k-1})+(a-b)+(x_{k-1}-b)+\dots+(x_1-b)} = \lambda^{k(a-b)}.$$

Note also that the reverse path \mathcal{P}^{-1} leading from s' to s has probability $\mathbf{P}(\mathcal{P}^{-1}) = \frac{\lambda^{k(b-a)}}{D(\mathcal{P})}$, where the denominator $D(\mathcal{P})$ is the same as above because all transitions in the chains are of the form $P(x,y) = N_{xy}/D_{xy}$ with $D_{xy} = D_{yx}$. Since \mathcal{M}_{adj} is reversible, we get the first of the following equalities:

$$\frac{\pi(s')}{\pi(s)} = \frac{\mathbf{P}(\mathcal{P})}{\mathbf{P}(\mathcal{P}^{-1})} = \frac{\lambda^{k(a-b)}}{\lambda^{k(b-a)}} = \frac{P^*(s,s')}{P^*(s',s)},$$

where P^* is the transition matrix of $\mathcal{M}_{\text{any}}^*$. Thus, the detailed balance condition for P^* is satisfied and π is the stationary distribution of $\mathcal{M}_{\text{any}}^*$. ■

In contrast to the other two chains, we show that \mathcal{M}_{any} is not reversible. This suggests that its stationary distribution is not easy to derive, and therefore also the comparison between \mathcal{M}_{any} and \mathcal{M}_{adj} is difficult.

Theorem 11. The chain \mathcal{M}_{any} is not reversible.

Proof. Consider three elements $a \leq b \leq c$, not all three equal. We say that an edge has *length* w if its transition probability is of the form $\frac{\lambda^w}{\lambda^w + \lambda^{-w}}$. The *length* of a path is the total length of its edges. The cycle $\mathcal{C} = (abc) \rightarrow (bac) \rightarrow (bca) \rightarrow (cba) \rightarrow (abc)$ in Fig. 3 has a length different from the reversed cycle, i.e., $\mathbf{P}(\mathcal{C}) \neq \mathbf{P}(\mathcal{C}^{-1})$. This violates the Kolmogorov reversibility criterion Eq. (3). ■

In our experiments (see Sec. VI), it turns out that doing only adjacent comparisons (\mathcal{M}_{adj}) is better than doing arbitrary comparisons (\mathcal{M}_{any}), in the sense that the expected weighted

number of inversions in their stationary distribution is lower for \mathcal{M}_{adj} than for \mathcal{M}_{any} ; the following sections provide analytical results for special cases. Note that Theorem 10 says that in the long-run $\mathcal{M}_{\text{any}}^*$ is as good as \mathcal{M}_{adj} .

IV. BINARY INPUTS

In this section we restrict to the case in which every element in the sequence is either a or b for some $b > a$. That is, the sorted sequence is $(a, \dots, a, b, \dots, b)$. We denote the number of a 's and b 's by n_a and n_b , respectively.

A. Mixing time

For binary inputs, we can uniquely express every sequence by a vector $v = (v_1, \dots, v_{n_b})$ with $v_1 \geq v_2 \geq \dots \geq v_{n_b}$, where $v_i \in \{0, \dots, n_a\}$ denotes the number of inversions of the i th b in the sequence (for example, *babba* corresponds to 211, while *babab* corresponds to 210). Such a vector is visualized as a monotonically decreasing “staircase” in a $n_b \times n_a$ grid and \mathcal{M}_{adj} corresponds to the biased Markov process in Ref. [35]. The bounds on the mixing time for this process translate immediately for our chain.

Theorem 12 (by Theorem 2.1 in Ref. [35]). For binary inputs, the mixing time of \mathcal{M}_{adj} satisfies

$$t_{\text{mix}}(\epsilon) = O(n^2 \log(\epsilon^{-1})).$$

Observe also that the chain \mathcal{M}_{adj} corresponds to the well-known ASEP (see, e.g., Refs. [10–16, 27, 28] for the analysis of the corresponding Markov chain).

We next consider \mathcal{M}_{any} and prove an upper bound. To bound the mixing time of \mathcal{M}_{any} we use the method of *path coupling* [36]. A *path coupling* for a chain \mathcal{M} can be specified by providing distributions

$$\mathbb{P}_{x,y}[X = x', Y = y'], \quad \text{for all } x, y \in S \text{ such that } P(x,y) > 0, \quad (5)$$

satisfying, for all $x, y \in S$ such that $P(x,y) > 0$,

$$\mathbb{P}_{x,y}[X = x'] = P(x, x') \quad \text{for all } x' \in S, \quad (6)$$

$$\mathbb{P}_{x,y}[Y = y'] = P(y, y') \quad \text{for all } y' \in S. \quad (7)$$

We use ρ to denote the shortest-path distance in the Markov chain, i.e., $\rho(x,y)$ is the minimum number of transitions to go from x to y .

Lemma 13 (Theorems 2.1 and 2.2 in Ref. [36]). Suppose there exists $\beta < 1$ such that, for all $x, y \in S$ with $P(x,y) > 0$, it holds that

$$\mathbb{E}_{x,y}[\rho(X,Y)] \leq \beta. \quad (8)$$

Then the mixing time $t_{\text{mix}}(\epsilon)$ of the Markov chain under consideration satisfies

$$t_{\text{mix}}(\epsilon) \leq \frac{\log(D\epsilon^{-1})}{1 - \beta},$$

where D is the maximum value that ρ achieves on $S \times S$.

Path coupling for \mathcal{M}_{any} . Consider two sequences x and y that differ by swapping elements in position i^* and j^* . For every such pair (x, y) we specify the probabilities in Eq. (5) to move to a pair (x', y') . We group the $\binom{n}{2}$ different swaps between elements in positions i and j as follows:

$$\begin{aligned} (i^*, j^*) &\leftrightarrow (i^*, j^*) \\ (i^*, j) &\leftrightarrow (j^*, j) \\ (i, j^*) &\leftrightarrow (i, i^*) \\ (i, j) &\leftrightarrow (i, j), \end{aligned}$$

in the sense that if we consider the positions i^* and j in one sequence, we consider the positions j^* and j in the other sequence, and vice versa. Clearly, this defines a bijection on the swaps of the two sequences. Now let $x_{i \times j}$ denote the sequence obtained from x by swapping the two elements at positions i and j . The path coupling is as follows:

$$\begin{aligned} \text{for } i^*, j^* \quad &(x, y) \mapsto (y, y) \\ &\text{with } P(x, y), \\ &(x, y) \mapsto (x, x) \\ &\text{with } P(y, x), \\ \text{for } i^*, j \quad &(x, y) \mapsto (x_{i^* \times j}, y_{i^* \times j}) \\ &\text{with } \min\{P(x, x_{i^* \times j}), P(y, y_{i^* \times j})\}, \\ &(x, y) \mapsto (x_{i^* \times j}, y) \\ &\text{with } \max\{0, P(x, x_{i^* \times j}) - P(y, y_{i^* \times j})\}, \\ &(x, y) \mapsto (x, y_{i^* \times j}) \\ &\text{with } \max\{0, P(y, y_{i^* \times j}) - P(x, x_{i^* \times j})\}, \\ \text{for } i, j^* \quad &(x, y) \mapsto (x_{i \times j^*}, y_{i \times j^*}) \\ &\text{with } \min\{P(x, x_{i \times j^*}), P(y, y_{i \times j^*})\}, \\ &(x, y) \mapsto (x_{i \times j^*}, y) \\ &\text{with } \max\{0, P(x, x_{i \times j^*}) - P(y, y_{i \times j^*})\}, \\ &(x, y) \mapsto (x, y_{i \times j^*}) \\ &\text{with } \max\{0, P(y, y_{i \times j^*}) - P(x, x_{i \times j^*})\}, \\ \text{for } i, j \quad &(x, y) \mapsto (x_{i \times j}, y_{i \times j}) \\ &\text{with } P(x, x_{i \times j}) = P(y, y_{i \times j}). \end{aligned}$$

Finally, with all remaining probability,

$$(x, y) \mapsto (x, y).$$

One can easily check that this is indeed a path coupling, that is, Eqs. (6) and (7) are satisfied. The difficulty is in proving the condition necessary to apply Lemma 13.

Lemma 14. Let $p = 1 - p_{ab}$. The path coupling defined above satisfies condition Eq. (8) with

$$\beta \leq 1 - \frac{2(1 + p(n-2))}{n(n-1)} \leq 1 - \frac{2p}{n-1}.$$

Proof. The second inequality follows from $p(n-2) + 1 \geq pn$, since $p \leq \frac{1}{2}$. We next prove the first inequality. Let x, y be two sequences that differ in swapping positions i^* and j^* , thus $\rho(x, y) = 1$. Since $P(x, y) + P(y, x) = 1$, the new distance $\rho(x', y')$ after choosing positions i^* and j^* is always zero. Furthermore, for every position k , such that $k \neq i^*$ and $k \neq j^*$, either $\rho(x_{i^* \times k}, y_{i^* \times k}) = 0$ or $\rho(x_{k \times j^*}, y_{k \times j^*}) = 0$ (see Fig. 3), and the probability of accepting such a transition is at least p . Note also that, in our coupling, each of these two transitions correspond to exactly one choice of the pair (i, j) of positions: in the first case $(i, j) = (i^*, k)$, and $(i, j) = (j^*, k)$ in the second case. Therefore, in either case, there are $n-2$ such pairs, one for each possible $k \notin \{i^*, j^*\}$. Finally, it is easy to see that after every other transition $\rho(x', y') = 1$. Since there are $\binom{n}{2}$ pairs of positions in a sequence, we conclude that $\mathbb{E}[\rho(x', y')] \leq (1 - \frac{1+p(n-2)}{\binom{n}{2}})$. ■

Theorem 15. Let $n' = \min\{n_a, n_b\} \leq \frac{n}{2}$ and $p = 1 - p_{ab}$. The mixing time of \mathcal{M}_{any} is

$$t_{\text{mix}}(\epsilon) \leq \frac{n(\log(n') - \log(\epsilon))}{2p}.$$

Proof. The maximum number of transitions required to go from any sequence to any other sequence is $D = \min\{n_a, n_b\} \leq \frac{n}{2}$: with D swaps we can either move all a or all b to their desired positions. By Lemmas 13 and 14, the claim follows immediately. ■

V. ONLY ADJACENT SWAPS IS BETTER

In this section, we prove for two special cases that in the long run the chain \mathcal{M}_{adj} comparing only adjacent elements is *better* than the chain \mathcal{M}_{any} performing comparisons between two arbitrary elements, where *better* means that the expected weighted number of inversion is smaller in \mathcal{M}_{adj} than in \mathcal{M}_{any} .

The first special case is when all elements are equal except for one. As a special case of binary sequences, the mixing time is implied. Additionally, we derive the stationary distribution for both \mathcal{M}_{adj} and \mathcal{M}_{any} and prove some relations between those two. In the second special case we leave binary sequences and consider sorting of three arbitrary elements (not all of them equal) and show that the probability of sorting correctly is higher in \mathcal{M}_{adj} than \mathcal{M}_{any} . We will also see that already for three elements, the analysis and derivation of the stationary distribution is difficult.

A. One outlier

We call *one outlier* the case in which we have $n-1$ small identical elements, and only one bigger element (the outlier) to be sorted. That is, the sorted sequence is (a, a, \dots, a, b) , with $b > a$. Since swapping two identical elements does not change the sequence, i.e., the state of the chain, we have n states which correspond to the possible positions of b . We denote the state in which b is in position i by $s^{(i)}$, so that $s^{(n)} = (a, \dots, a, b)$ is the sorted sequence and $s^{(1)} := (b, a, \dots, a)$ is the “reversely sorted” sequence. Note that the weighted number of inversions of $s^{(i)}$ is $(n-i)(b-a)$, thus the expected weighted number of

inversions is

$$E^w := \sum_{i=1}^n (n-i)(b-a)\pi(s^{(i)}). \quad (9)$$

It is useful for the analysis to consider the probability that element $b > a$ is erroneously declared smaller than a in a single comparison,

$$p := 1 - p_{ab} = \frac{\lambda^{a-b}}{\lambda^{a-b} + \lambda^{b-a}}.$$

Observation 1. In the one outlier case, the chain \mathcal{M}_{adj} becomes a so-called birth-and-death chain meaning that from each state $s^{(i)}$ we can only move to $s^{(i+1)}$ or to $s^{(i-1)}$, and the transition probabilities are $P(s^{(i+1)}, s^{(i)}) = \frac{p}{n-1}$ and $P(s^{(i)}, s^{(i+1)}) = \frac{1-p}{n-1}$, for all $i \in \{1, \dots, n-1\}$. In the chain \mathcal{M}_{any} every state is connected to all other states and the transition probabilities are $P(s^{(i)}, s^{(j)}) = p/\binom{n}{2}$ if $i > j$ and $(1-p)/\binom{n}{2}$ if $i < j$.

Lemma 17. The stationary distributions of \mathcal{M}_{adj} and \mathcal{M}_{any} are

$$\pi_{\text{adj}}(s^{(i)}) = \frac{p^{n-i}(1-p)^{i-1}(1-2p)}{(1-p)^n - p^n},$$

$$\pi_{\text{any}}(s^{(i)}) = \frac{np(1-p)}{[(n-i+1)(1-p) + (i-1)p][(n-i)(1-p) + ip]}.$$

Lemma 18. The expected weighted number of inversions for \mathcal{M}_{adj} and \mathcal{M}_{any} are

$$E_{\text{adj}}^w = n(b-a)p \left[\frac{1}{n(1-2p)} - \frac{p^{n-1}}{(1-p)^n - p^n} \right] < (b-a) \frac{p}{1-2p},$$

$$E_{\text{any}}^w = n(b-a)p \sum_{i=0}^{n-1} \frac{i(1-p)}{[(i+1)(1-p) + (n-i-1)p][i(1-p) + (n-i)p]} > n(b-a)p.$$

Finally, we conclude a corollary that compares the stationary distributions of two states in \mathcal{M}_{adj} and \mathcal{M}_{any} .

Corollary 19. For any two states s and s' that differ in exactly one adjacent swap, if the weighted number of inversions satisfies $w(s') > w(s)$, then it holds that $\frac{\pi_{\text{adj}}(s')}{\pi_{\text{adj}}(s)} < \frac{\pi_{\text{any}}(s')}{\pi_{\text{any}}(s)}$.

We will see in the next section, that this equation not only holds for one outlier sequences of arbitrary length, but also for sequences containing three arbitrary elements (Lemma 21).

B. Three arbitrary elements

Our second special case is to consider sorting three arbitrary elements and to show that \mathcal{M}_{adj} has more chances to return the sorted sequence than \mathcal{M}_{any} .

Theorem 20. For any three elements, not all of them identical, the chain \mathcal{M}_{adj} returns the sorted sequence with a probability (at stationary distribution) strictly larger than that of \mathcal{M}_{any} (at stationary distribution),

$$\pi_{\text{adj}}(abc) > \pi_{\text{any}}(abc) \quad (10)$$

where abc is the sorted sequence ($a \leq b \leq c$), for all $\lambda > 0$.

To prove this theorem we show that the ratios between the distribution of adjacent states in \mathcal{M}_{adj} gets “worse” in \mathcal{M}_{any} :

The next theorem says that the chain \mathcal{M}_{adj} has a better probability of returning the sorted sequence and a better expected weighted number of inversions than \mathcal{M}_{any} .

Theorem 16. For the case of one outlier, the following holds. The probability of obtaining the sorted sequence, at stationary distribution, is constant for the \mathcal{M}_{adj} , while for \mathcal{M}_{any} it converges to zero as n grows:

$$\pi_{\text{adj}}(s^{(\text{sorted})}) > \frac{1-2p}{1-p}, \quad \pi_{\text{any}}(s^{(\text{sorted})}) < \frac{1-p}{np}.$$

The expected weighted number of inversions is constant for \mathcal{M}_{adj} and linear in n for \mathcal{M}_{any} :

$$E_{\text{adj}}^w < \frac{p}{1-2p}(b-a), \quad E_{\text{any}}^w > np(b-a).$$

The theorem above follows immediately from the next two lemmas, which we will state here and prove in Appendix A. Recall that $s^{(\text{sorted})} = s^{(n)}$.

Lemma 21. For any two states s and s' that differ in exactly one adjacent swap, if the weighted number of inversions satisfies $w(s') > w(s)$, then it holds that

$$\frac{\pi_{\text{adj}}(s')}{\pi_{\text{adj}}(s)} < \frac{\pi_{\text{any}}(s')}{\pi_{\text{any}}(s)}. \quad (11)$$

Proof Idea. We use the Markov Chain Tree Theorem Eq. (4). Our goal is thus to show that

$$\frac{\sum_{T \in \mathcal{T}(s)} \mathbf{P}(T)}{\sum_{T' \in \mathcal{T}(s')} \mathbf{P}(T)} < \frac{\pi_{\text{adj}}(s)}{\pi_{\text{adj}}(s')} = \lambda^{2(w(s')-w(s))} \quad \text{for } w(s) < w(s'). \quad (12)$$

Ideally, one would like to find a bijection from trees $T \in \mathcal{T}(s)$ to trees $T' \in \mathcal{T}(s')$ such that $\frac{\mathbf{P}(T)}{\mathbf{P}(T')} < \frac{\pi_{\text{adj}}(s)}{\pi_{\text{adj}}(s')}$ holds for each tree $T \in \mathcal{T}(s)$. Unfortunately, this is in general not possible, so the following slightly more involved argument is used:

(1) The simple mapping we use consists in reversing the path from s' to s in T to obtain the new tree T' . This mapping is a bijection between $\mathcal{T}(s)$ and $\mathcal{T}(s')$.

(2) Because this mapping does not guarantee the desired inequality for all trees T , we classify the trees in $\mathcal{T}(s)$ into *good* and *bad* trees: a tree T is *bad* if $\frac{\mathbf{P}(T)}{\mathbf{P}(T')} > \lambda^{2(w(s')-w(s))}$ and

good otherwise. We then show that

$$\frac{\sum_{T \in \text{bad}} \mathbf{P}(T) + \sum_{T \in \text{good}} \mathbf{P}(T)}{\sum_{T \in \text{bad}} \mathbf{P}(T') + \sum_{T \in \text{good}} \mathbf{P}(T')} < \lambda^{2[w(s') - w(s)]},$$

where T' is the tree obtained from T via the mapping in the previous item. This proves Eq. (12) since *good* and *bad* define a partition of $\mathcal{T}(s)$ and also a partition of $\mathcal{T}(s')$.

The details of this proof are given in Appendix B.

From this lemma it is easy to obtain Eq. (10) in Theorem 20.

Proof of Theorem 20. By transitivity, Lemma 21 implies that, for all nonsorted sequences $s \neq (abc)$, $\frac{\pi_{\text{adj}}(s)}{\pi_{\text{adj}}(abc)} < \frac{\pi_{\text{any}}(s)}{\pi_{\text{any}}(abc)}$, and therefore

$$\begin{aligned} \pi_{\text{adj}}(abc) &= \frac{1}{1 + \sum_{s \neq (abc)} \frac{\pi_{\text{adj}}(s)}{\pi_{\text{adj}}(abc)}} > \frac{1}{1 + \sum_{s \neq (abc)} \frac{\pi_{\text{any}}(s)}{\pi_{\text{any}}(abc)}} \\ &= \pi_{\text{any}}(abc). \end{aligned}$$

VI. EXPERIMENTAL EVALUATION

We conducted a set of experiments on several input sequences to compare the three sorting algorithms (chains \mathcal{M}_{adj} , \mathcal{M}_{any} , and $\mathcal{M}_{\text{any}}^*$). In our experiments, we consider the following aspects:

(1) *Low energy (high noise) regime.* We evaluate how much the first two algorithms are robust to an increase of the error probability by taking $\lambda = e^{1/\text{noise}}$ for increasing values of noise.

We estimate the expected weighted numbers of inversions of \mathcal{M}_{adj} and \mathcal{M}_{any} by computing the average value of a set of samples after both chains have converged to a stable distribution. The results in Fig. 4 show that \mathcal{M}_{any} degrades much earlier than \mathcal{M}_{adj} .

(2) *Evolution to stationary distribution.* We compare all three algorithms \mathcal{M}_{adj} , \mathcal{M}_{any} , and $\mathcal{M}_{\text{any}}^*$ in Fig. 5. These experiments suggest that, for some intermediate range of noise values, $\mathcal{M}_{\text{any}}^*$ possesses good features from both the other algorithms: the weighted number of inversions decreases faster than \mathcal{M}_{adj} , while its stationary distribution is of course better than \mathcal{M}_{any} .

(3) *Probability of getting sorted sequence.* We evaluate how the probability of hitting the sorted sequence changes when the noise increases.

For this, we count the numbers of hits in a set of samples after both chains have converged to a stable distribution. Figure 6 deals with the sequence of ten different elements $\{1, 2, \dots, 10\}$, while Fig. 7 is about the one outlier $\{1, 1, 1, 1, 1, 1, 1, 1, 2\}$. In both cases and for all values of noise, \mathcal{M}_{adj} has a higher hitting rate of the sorted sequence than \mathcal{M}_{any} .

We will discuss these three types of experiments in the three subsections below. To sample from the stationary distribution we need two things: (i) the chain needs to be mixed, i.e., converged to its stationary distribution, and (ii) the samples need to be independent.

(i) *Convergence time.* We consider a chain as *converged* as soon as the averaged weighted number of inversions over intervals of samples does not change (decrease further, when started with the reverse order). Technically, this is not equivalent to waiting until the chains are mixed, but is exactly what we need if we are interested in the quality of the solutions.

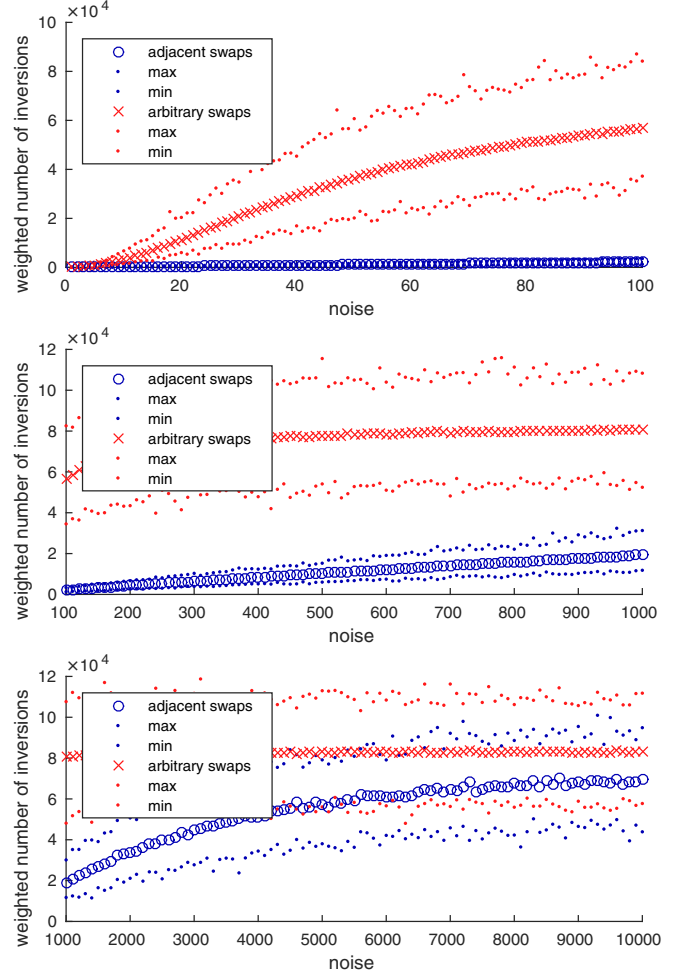


FIG. 4. A comparison of \mathcal{M}_{adj} (adjacent swaps) and \mathcal{M}_{any} (arbitrary swaps) on the long run. We measure the average, maximum, and minimum weighted number of inversions for a certain number of steps after both algorithms have approached their stationary distribution. The elements to be sorted are $(100, 199, \dots, 1)$. From top to bottom: The first graph shows the evolution of the weighted number of inversions for increasing noise, where $\lambda = e^{1/\text{noise}}$, from 0 to 100. Similarly, in the second graph the noise is increased from 100 to 1000 and in the third graph from 1000 to 10 000. The graphs indicate that by increasing the value of noise, the stationary behavior of \mathcal{M}_{any} degrades much earlier than that of \mathcal{M}_{adj} .

(ii) *Sampling from stationary distribution.* Ideally, one would start the chain anew for each sample and wait until it is mixed. However, for the stationary distribution it does not matter from which sequence the chain starts, thus we can instead continue the chain for a sufficient number of steps before sampling the next sequence.

A. Low-energy (high-noise) regime

For increasing values of error probabilities, we measure the average, maximum, and minimum weighted number of inversions for a certain number of steps after both algorithms \mathcal{M}_{adj} and \mathcal{M}_{any} have approached their stationary distribution.

We start with a reversed sequence and let the chains evolve until their average weighted number of inversions does not

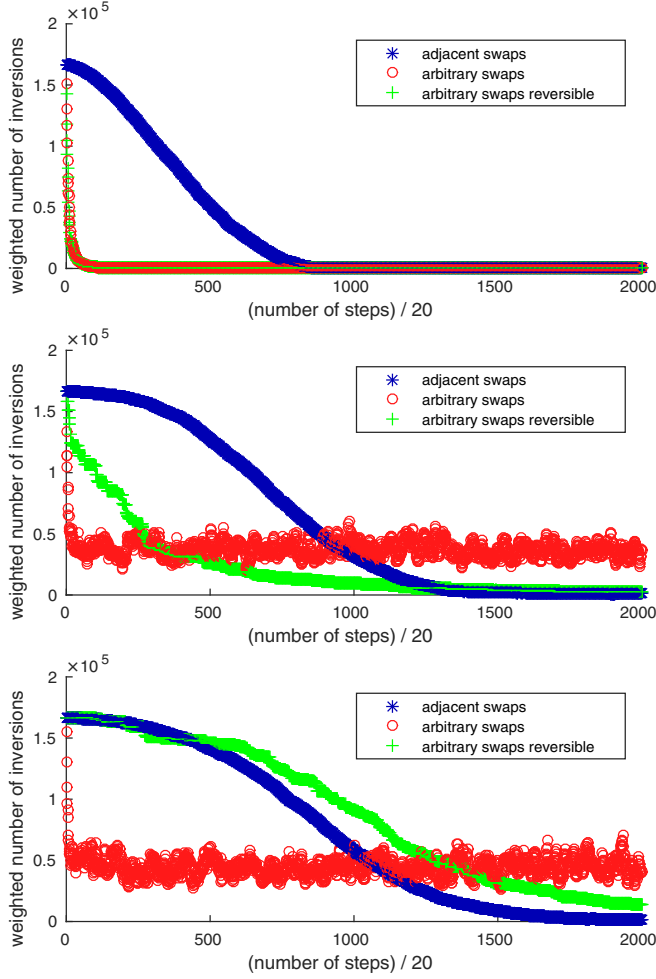


FIG. 5. A comparison of the three algorithms \mathcal{M}_{adj} (adjacent swaps), \mathcal{M}_{any} (arbitrary swaps), and $\mathcal{M}_{\text{any}}^*$ (arbitrary swaps reversible). For three different values of noise, we plot for each algorithm the evolution of the weighted number of inversions starting from the sequence $(100, 99, \dots, 1)$. After every 20 steps of the algorithm, the weighted number of inversions is plotted. *From top to bottom:* The plots are for the noise values 0.5, 50, and 65, where $\lambda = e^{1/\text{noise}}$. The three graphs indicate that for some values of noise, $\mathcal{M}_{\text{any}}^*$ possesses good features from the other two algorithms: the weighted number of inversions decreases faster than \mathcal{M}_{adj} , while its stationary distribution is of course better than \mathcal{M}_{any} , which is evident in the second plot. Note that the number of steps matches the number of comparisons made in \mathcal{M}_{adj} and \mathcal{M}_{any} , but not in $\mathcal{M}_{\text{any}}^*$.

change anymore: we sample every $R = \Theta(n^2)$ th sequence and compute the average weighted number of inversions avg of $T = \Theta(n)$ such consecutive samples. Then we compute the average avg_new of the next T samples and compare it with avg . If $|\frac{\text{avg_new}}{\text{avg}} - 1| \leq c$, where $0 < c < 1$ is a small constant, we say the chain is converged; otherwise, we repeat. After convergence, we sample again every $R' = \Theta(n^2)$ th sequence and consider $T' = \Theta(n)$ samples. Out of these, we compute the average, maximum, and minimum weighted number of inversions.

We repeat this experiment for different values of noise, where $\lambda = e^{1/\text{noise}}$ is the energy parameter. The experiments suggest that by increasing the value of noise, the stationary

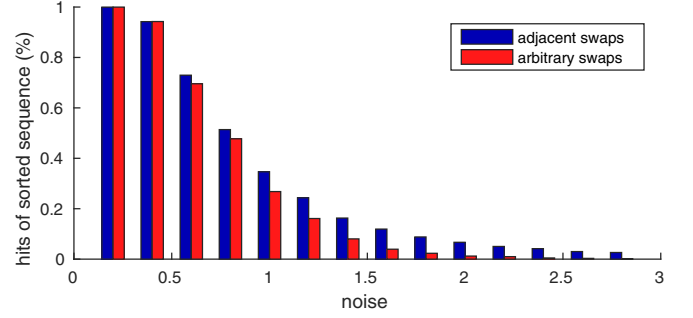


FIG. 6. A comparison of the two algorithms \mathcal{M}_{adj} (adjacent swaps) and \mathcal{M}_{any} (arbitrary swaps). For each algorithm and for increasing values of noise, we measure the probability of hitting the sorted sequence among a certain number of samples after both algorithms have converged. The noise increases from 0.2 to 2.8, where $\lambda = e^{1/\text{noise}}$, the initial sequence is $(10, 9, \dots, 1)$. The plot indicates that, for increasing values of noise, the stationary probability of the sorted sequence decreases faster in \mathcal{M}_{any} than in \mathcal{M}_{adj} .

behavior (average weighted number of inversions) of \mathcal{M}_{any} degrades much earlier than that of \mathcal{M}_{adj} .

In Fig. 4, we plot the graphs for the initial sequence $(100, 99, \dots, 1)$. We use parameters $R = 100$, $T = 10\,000$, $c = 0.05$, $R' = 1000$, and $T' = 20\,000$. In the three subfigures, we consider three different ranges of noise values: from 0 to 100, from 100 to 1000, and from 1000 to 10 000.

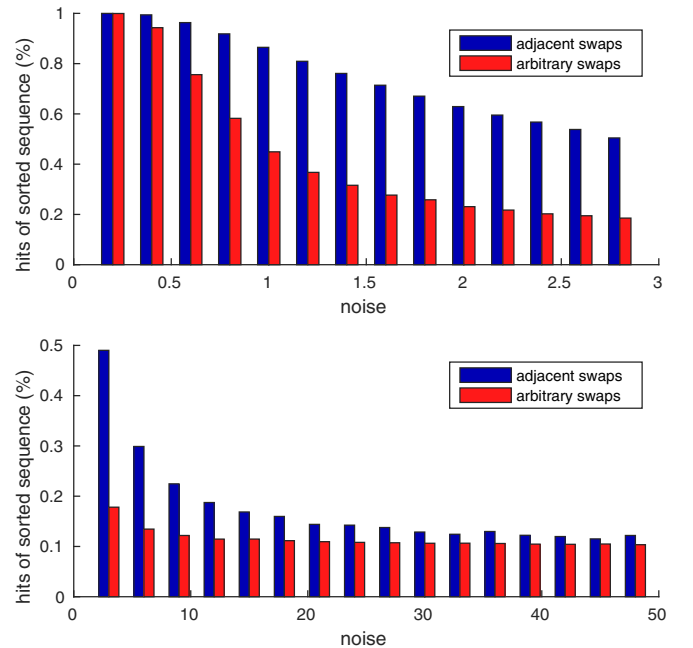


FIG. 7. A comparison of the two algorithms \mathcal{M}_{adj} (adjacent swaps) and \mathcal{M}_{any} (arbitrary swaps) on binary sequences. For each algorithm and for increasing values of noise, we measure the percentage of hits of the sorted sequence after both algorithms have reached their stationary distribution. The set of elements is $\{1, 1, 1, 1, 1, 1, 1, 1, 2\}$. *From top to bottom:* In the first plot, the values of the noise lie between 0 and 3, where $\lambda = e^{1/\text{noise}}$. In the second plot, the values of the noise lie between 3 and 50. The plots show that, for increasing values of noise, the stationary probability of the sorted sequence decreases much faster and is always lower in \mathcal{M}_{any} than in \mathcal{M}_{adj} .

B. Evolution to stationary distribution

In this experiment, we observe the evolution of the weighted number of inversions for all three chains \mathcal{M}_{adj} , \mathcal{M}_{any} , and $\mathcal{M}_{\text{any}}^*$. In particular, we start from the reversed order and wait until all chains have converged their stationary distributions. This experiment is repeated for different values of noise, where $\lambda = e^{1/\text{noise}}$ is the energy parameter.

We observe that for very small values of noise, the expected weighted numbers of inversions are almost the same for \mathcal{M}_{any} and \mathcal{M}_{adj} or $\mathcal{M}_{\text{any}}^*$. Moreover, also the time to converge is the same for the three chains. For an intermediate range of noise values, the experiments suggest that, $\mathcal{M}_{\text{any}}^*$ is a good compromise between the other two algorithms: the weighted number of inversions decreases faster than \mathcal{M}_{adj} , while its stationary distribution is of course better than \mathcal{M}_{any} . However, for high values of noise, $\mathcal{M}_{\text{any}}^*$ converges even slower than \mathcal{M}_{adj} . This can be explained with the observation that the probability of performing a swap (at all) in $\mathcal{M}_{\text{any}}^*$ becomes very small if the energy parameter tends to 1. For almost every pair of elements in a sequence of n elements, its distance is in $O(n)$. Hence, a comparison of this pair is repeated $O(n)$ times and the probability that all outcomes are the same tends to $\frac{1}{2^n}$.

Remember also, that the time for one step of $\mathcal{M}_{\text{any}}^*$ can be much higher than for \mathcal{M}_{adj} or \mathcal{M}_{any} , since it requires several comparisons, i.e., the number of comparisons can be linear in the length of the sequence.

In Fig. 5, we plot for each algorithm the evolution of the weighted number of inversions starting from the sequence (100,99,...,1). After every 20 steps of the algorithm, the weighted number of inversions is plotted. We illustrate the experiment for three different values of noise, namely 0.5, 50, and 65.

C. Probability of getting sorted sequence

In this experiment, we compare the stationary probability of the sorted sequence of \mathcal{M}_{adj} and \mathcal{M}_{any} . We let the chains evolve until their average number of weighted inversions does not change anymore (see Sec. VIA for the details). After convergence, we sample every $\Theta(n^2)$ th sequence and consider $\Theta(n^2)$ samples. Among those, we count the number of hits of the sorted sequence. We repeat this experiment for different values of noise, where $\lambda = e^{1/\text{noise}}$. The experiment indicates that, for increasing values of noise, the stationary probability of the sorted sequence decreases much faster in \mathcal{M}_{any} than in \mathcal{M}_{adj} .

In Fig. 6, we show the plot for the sequence of ten different elements $\{1, 2, \dots, 10\}$. The chains are converged after 10 000 steps. The number of samples is 10 000 and we wait 1000 steps before the next sample. We let the noise increase from 0.2 to 2.8.

In Fig. 7 we illustrate the experiment for the one outlier sequence $\{1, 1, 1, 1, 1, 1, 1, 1, 2\}$. Again, the chains are converged after 10 000 steps. The number of samples is 10 000 and we wait 1000 steps before the next sample. We let the noise increase from 0.2 to 2.8 in the first plot and from 3 to 50 in the second plot.

We chose a small number of elements, since otherwise the probability of hitting is already very small and the differences are hardly distinguishable by eyes.

VII. CONCLUSION AND OPEN QUESTIONS

In this work we initiated the *compared analysis* of simple sorting processes (algorithms) based on repeated “compare and swap” operations between pairs of elements. In particular, we observed experimentally that *adjacent swaps* perform *better* than *arbitrary swaps* in terms of sortedness of the sequence, once the two processes have reached their respective equilibrium. In all of our experiments, the same behavior occurs at different values of $\lambda > 1$ and for different input sequences (see Sec. VI for details).

Our main technical contribution is to provide a rigorous analysis of these processes in some restricted cases (see Table I). In particular, we have studied their *stationary distributions* and *mixing times*: The former gives us the “sortedness” of the sequence after “sufficiently many” comparisons, and the latter gives us the number of such comparisons. For the reversible process \mathcal{M}_{adj} with adjacent swaps, we are able to characterize its stationary distribution in the general case.

The main obstacle in the analysis seems the lack of general results on *nonreversible* Markov chains. For instance, nonreversible chains lead to the *lifting technique* to provide *faster* versions of reversible chains [37,38]. The technique consists of constructing a nonreversible chain which has the same stationary distribution as a reversible. In our problem, instead, we want to show that two given chains have *significantly different* distributions. Also, other comparison techniques to bound the mixing time in Markov chains are based on the assumption that the chains are reversible [39].

The main question left open is to determine the stationary distribution for the chain \mathcal{M}_{any} performing *arbitrary swaps*, or at least an estimate of such probabilities for arbitrary input sequences. Here we tackled the case of (1) binary inputs with arbitrarily many elements and (2) three arbitrary elements. These results support the intuition that what we see in Fig. 1 is not accidental. Perhaps more importantly, they give an indication on the tools and on the difficulties towards the general case. Already for binary inputs with more than one outlier, results on sortedness are not straightforward, since the chain is closely related to *multidimensional birth-and-death chains* whose analysis is highly nontrivial, even in the two-dimensional case [40]; the latter would relate to the *two outliers* in our problem.

We also find it intriguing to compare the mixing time of the three chains \mathcal{M}_{adj} , \mathcal{M}_{any} , and $\mathcal{M}_{\text{any}}^*$. Recall that we proved that $\mathcal{M}_{\text{any}}^*$ has the same stationary distribution of \mathcal{M}_{adj} . Our experiments suggest that, for some values of $\lambda > 1$, $\mathcal{M}_{\text{any}}^*$ might be a good compromise between \mathcal{M}_{adj} and \mathcal{M}_{any} , as it is faster than \mathcal{M}_{adj} but slower than \mathcal{M}_{any} in terms of number of steps. However, this picture of the processes is not entirely fair since one step of $\mathcal{M}_{\text{any}}^*$ involves multiple comparisons, while only one comparison per step is made in the other two processes. These multiple comparisons also explain the observation that, for small values of λ (high noise, low energy), $\mathcal{M}_{\text{any}}^*$ converges even slower to its stationary distribution than \mathcal{M}_{adj} , since swapping two elements is only possible if all comparisons yield the same result.

An extension to adjacent and any pairs swaps would be to parametrize the maximum distance of swaps, i.e., allow only swaps between elements whose positions differ by at most a threshold k where $1 \leq k \leq n$. For constant biased comparisons

(every comparison is correct with probability $p > 1/2$), it has been shown that in such a case the expected weighted number of inversions lies between the two extremal cases (adjacent and arbitrary swaps) [30].

ACKNOWLEDGMENTS

We are grateful to Lucas Boczkowski, Tomáš Gavenčíak, Francesco Pasquale, and Peter Widmayer for useful

discussions. Research supported by SNF (Project No. 200021_165524, Energy-Tunable Combinatorial Algorithms).

APPENDIX A: PROOFS LEMMAS 17 AND 18 (ONE OUTLIER)

In this Appendix, we will prove Lemmas 17 and 18 from Sec. V A.

Proof of Lemma 17. We get the stationary distribution for \mathcal{M}_{adj} using the global balance condition of stationary distribution,² which implies that $\pi_{\text{adj}}(s^{(i)}) = \pi_{\text{adj}}(s^{(i+1)})\left(\frac{1-p}{p}\right)$. By the structure of our Markov chain, it holds for any state $s^{(j)}$ that

$$\pi_{\text{adj}}(s^{(j)}) = \pi_{\text{adj}}(s^{(i)}) \left(\frac{1-p}{p}\right)^{i-j}.$$

Since the sum of all π_{adj} is one, we can express $\pi_{\text{adj}}(s^{(i)})$ as

$$\begin{aligned} \pi_{\text{adj}}(s^{(i)}) &= 1 - \pi_{\text{adj}}(s^{(i)}) \left[\sum_{j=1}^{i-1} \left(\frac{p}{1-p}\right)^j + \sum_{j=1}^{n-i} \left(\frac{1-p}{p}\right)^j \right] \\ &= \frac{1}{1 + \sum_{j=1}^{i-1} \left(\frac{p}{1-p}\right)^j + \sum_{j=1}^{n-i} \left(\frac{1-p}{p}\right)^j}. \end{aligned}$$

By applying the formula for geometric series, we rewrite the above equation and then expand all terms to the same denominator:

$$\begin{aligned} \frac{1}{\pi_{\text{adj}}(s^{(i)})} &= 1 + \frac{1 - \left(\frac{p}{1-p}\right)^i}{1 - \left(\frac{p}{1-p}\right)} - 1 + \frac{1 - \left(\frac{1-p}{p}\right)^{n-i+1}}{1 - \left(\frac{1-p}{p}\right)} - 1 \\ &= \frac{-(1-2p)(1-p)^{i-1}p^{n-i} + (1-p)^i p^{n-i} - p^n - (1-p)^{i-1}p^{n-i+1} + (1-p)^n}{(1-2p)(1-p)^{i-1}p^{n-i}}. \end{aligned}$$

Finally, we use that $(1-2p) = (1-p) - p$ and observe that the fraction simplifies to what we claimed.

The formula for the stationary distribution of \mathcal{M}_{any} is also derived using the global balance condition of stationary distribution. For any state $s^{(i)}$ it holds that

$$\begin{aligned} \pi_{\text{any}}(s^{(i)})[(i-1)p + (n-i)(1-p)] &= (1-p) \sum_{j=1}^{i-1} \pi_{\text{any}}(s^{(j)}) + p \sum_{j=i+1}^n \pi_{\text{any}}(s^{(j)}) \\ &= (1-p) \sum_{j=1}^{i-1} \pi_{\text{any}}(s^{(j)}) + p \left\{ 1 - \sum_{j=1}^i \pi_{\text{any}}(s^{(j)}) \right\} \\ \pi_{\text{any}}(s^{(i)})[ip + (n-i)(1-p)] &= (1-2p) \sum_{j=1}^{i-1} \pi_{\text{any}}(s^{(j)}) + p. \end{aligned}$$

By induction on i we show that this recurrence resolves to what we claimed. For $i=1$ we immediately get $\pi_{\text{any}}(s^{(1)})[p + (n-1)(1-p)] = p$. By multiplying both sides by $n(1-p)$, the claim follows trivially. For $i > 1$ we assume that the recursive formula holds for $i-1$ and we get

$$\begin{aligned} \pi_{\text{any}}(s^{(i)}) &= \frac{(1-2p) \sum_{j=1}^{i-1} \pi_{\text{any}}(s^{(j)}) + p}{[ip + (n-i)(1-p)]} = \frac{(1-2p) \sum_{j=1}^{i-2} \pi_{\text{any}}(s^{(j)}) + p + \pi_{\text{any}}(s^{(i-1)})(1-2p)}{[ip + (n-i)(1-p)]} \\ &= \frac{\pi_{\text{any}}(s^{(i-1)})[(i-1)p + (n-i+1)(1-p)] + \pi_{\text{any}}(s^{(i-1)})(1-2p)}{[ip + (n-i)(1-p)]} \\ &= \frac{np(1-p)}{[(n-i+1)(1-p) + (i-1)p][ip + (n-i)(1-p)]}. \end{aligned}$$

■

²The stationary distribution π of a Markov chain with transition matrix P must satisfy $\sum_{s' \neq s} \pi(s)P(s, s') = \sum_{s' \neq s} \pi(s')P(s', s)$, for any two states s, s' .

Proof of Lemma 18. We apply the generic formula Eq. (9) to derive the expected weighted number of inversions of \mathcal{M}_{adj} and \mathcal{M}_{any} . Since $\pi_{\text{adj}}(s^{(i)}) = \pi_{\text{adj}}(s^{(n)}) \left(\frac{p}{1-p}\right)^{n-i}$ we get

$$\begin{aligned} E_{\text{adj}}^w &= \sum_{i=1}^n (n-i)(x-1)\pi_{\text{adj}}(s^{(i)}) \\ &= (x-1)\pi_{\text{adj}}(s^{(n)}) \sum_{i=0}^{n-1} i \left(\frac{p}{1-p}\right)^i \\ &= (x-1) \frac{(1-p)^{n-1}(1-2p)}{(1-p)^n - p^n} \frac{(n-1)\left(\frac{p}{1-p}\right)^{n-1} - n\left(\frac{p}{1-p}\right)^n + \left(\frac{p}{1-p}\right)}{\left(\frac{p}{1-p} - 1\right)^2} \\ &= n(x-1)p \left[\frac{1}{n(1-2p)} - \frac{p^{n-1}}{(1-p)^n - p^n} \right]. \end{aligned}$$

Since $0 < p < \frac{1}{2}$, the first inequality is immediate from $\frac{p^n}{(1-p)^n - p^n} > 0$. For \mathcal{M}_{any} we have

$$\begin{aligned} E_{\text{any}}^w &= \sum_{i=1}^n (n-i)(x-1)\pi_{\text{any}}(s^{(i)}) = (x-1) \sum_{i=0}^{n-1} i \pi_{\text{any}}(s^{(n-i)}) \\ &= n(x-1)p \sum_{i=0}^{n-1} \frac{i(1-p)}{[(i+1)(1-p) + (n-i-1)p][i(1-p) + (n-i)p]}. \end{aligned}$$

Observe that we can lower bound the sum in the formula by the following integral, which is larger than 1 if $0 < p < \frac{1}{2}$:

$$\int_0^n \frac{i(1-p)}{[(i+1)(1-p) + (n-i-1)p][i(1-p) + (n-i)p]} di.$$

■

APPENDIX B: PROOF OF LEMMA 21 (THREE ARBITRARY ELEMENTS)

In this Appendix, we prove Lemma 21 from Sec. VB.

We shall use the Markov Chain Tree Theorem Eq. (4). Our goal is thus to show that

$$\begin{aligned} &\frac{\sum_{T \in \mathcal{T}(s)} \mathbf{P}(T)}{\sum_{T' \in \mathcal{T}(s')} \mathbf{P}(T')} \\ &< \frac{\pi_{\text{adj}}(s)}{\pi_{\text{adj}}(s')} = \lambda^{2(w(s')-w(s))} \quad \text{for } w(s) < w(s'). \end{aligned} \quad (\text{B1})$$

The strategy to prove this inequality is to find a suitable mapping from trees $T \in \mathcal{T}(s)$ to trees $T' \in \mathcal{T}(s')$ such that $\frac{\mathbf{P}(T)}{\mathbf{P}(T')} < \frac{\pi_{\text{adj}}(s)}{\pi_{\text{adj}}(s')}$. The basic mapping consists of reversing the path from s' to s in T to obtain the tree T' :

Definition 22. For any s and s' , and for any tree $T \in \mathcal{T}(s)$ its s' -reversed is the tree T' equal to T but with the edges on the path from s' to s reversed.

Hereafter, we call such a tree T' simply a *reversed* tree of T if s and s' are clear from the context. We say that an edge has *length* w if its transition probability is of the form $\frac{\lambda^w}{\lambda^w + \lambda^{-w}}$. The *length* of a path is the total length of its edges. The *multiset* of a tree T is the set of absolute edge lengths appearing in T ,

$$ms(T) = \{w \mid \text{some edge in } T \text{ has length } w \text{ or } -w\}.$$

Based on this multiset, we denote by $ms(T, w)$ the number of edges in T whose length is either w or $-w$.

Fact 1. Let T be a tree containing a path from s' to s of length ℓ , and let T' be its reversed tree. Then the probabilities

of these two trees are of the form

$$\mathbf{P}(T) = \frac{\lambda^\ell \lambda^{L(T)}}{M(T)} \quad \text{and} \quad \mathbf{P}(T') = \frac{\lambda^{-\ell} \lambda^{L(T)}}{M(T)}, \quad (\text{B2})$$

where $L(T)$ denotes the total length of the edges which are in the tree but not in the path between s and s' , and $M(T)$ depends on the multiset $ms(T)$, i.e.,

$$M(T) = \prod_{w \in ms(T)} (\lambda^w + \lambda^{-w})^{ms(T, w)}.$$

From Eq. (B2) we get $\frac{\mathbf{P}(T)}{\mathbf{P}(T')} \leq \lambda^{2\ell}$. This motivates the following definition.

Definition 23. For any s and s' , we call a tree $T \in \mathcal{T}(s)$ *good* if the length ℓ of its path from s' to s satisfies $\lambda^{2\ell} \leq \frac{\pi_{\text{adj}}(s)}{\pi_{\text{adj}}(s')}$. Otherwise, we call T a *bad* tree.

By this definition the basic mapping of a bad tree does not give the desired bound Eq. (B1). In such cases, we will map groups of bad trees into groups of good ones, depending on s and s' .

The (bac) versus (bca) case. Observe that $\frac{\pi_{\text{adj}}(\text{bac})}{\pi_{\text{adj}}(\text{bca})} = \lambda^{2(c-a)}$. In this case there is no bad tree, since all paths from (bca) to (bac) have length at most $c-a$ (see Fig. 8).

The (abc) versus (bac) case. Note that $\frac{\pi_{\text{adj}}(\text{bac})}{\pi_{\text{adj}}(\text{bca})} = \lambda^{2(b-a)}$. Figure 9 shows all paths from $s' = (\text{bac})$ to $s = (\text{abc})$. Trees using the path in Fig. 9(d) are bad, since this path has length $\ell = c-a > b-a$. Trees using the path in Fig. 9(c) are bad if $c-b > b-a$. We combine the bad trees with the good trees [Figs. 9(b), 9(f), 9(h), and 9(i)].

Lemma 24. Let $\mathcal{BAD}(s) \subset \mathcal{T}(s)$ be the set of bad trees with s' - s -path Figs. 9(c) or 9(d), and let $\mathcal{GOOD}(s) \subset \mathcal{T}(s)$ be the set

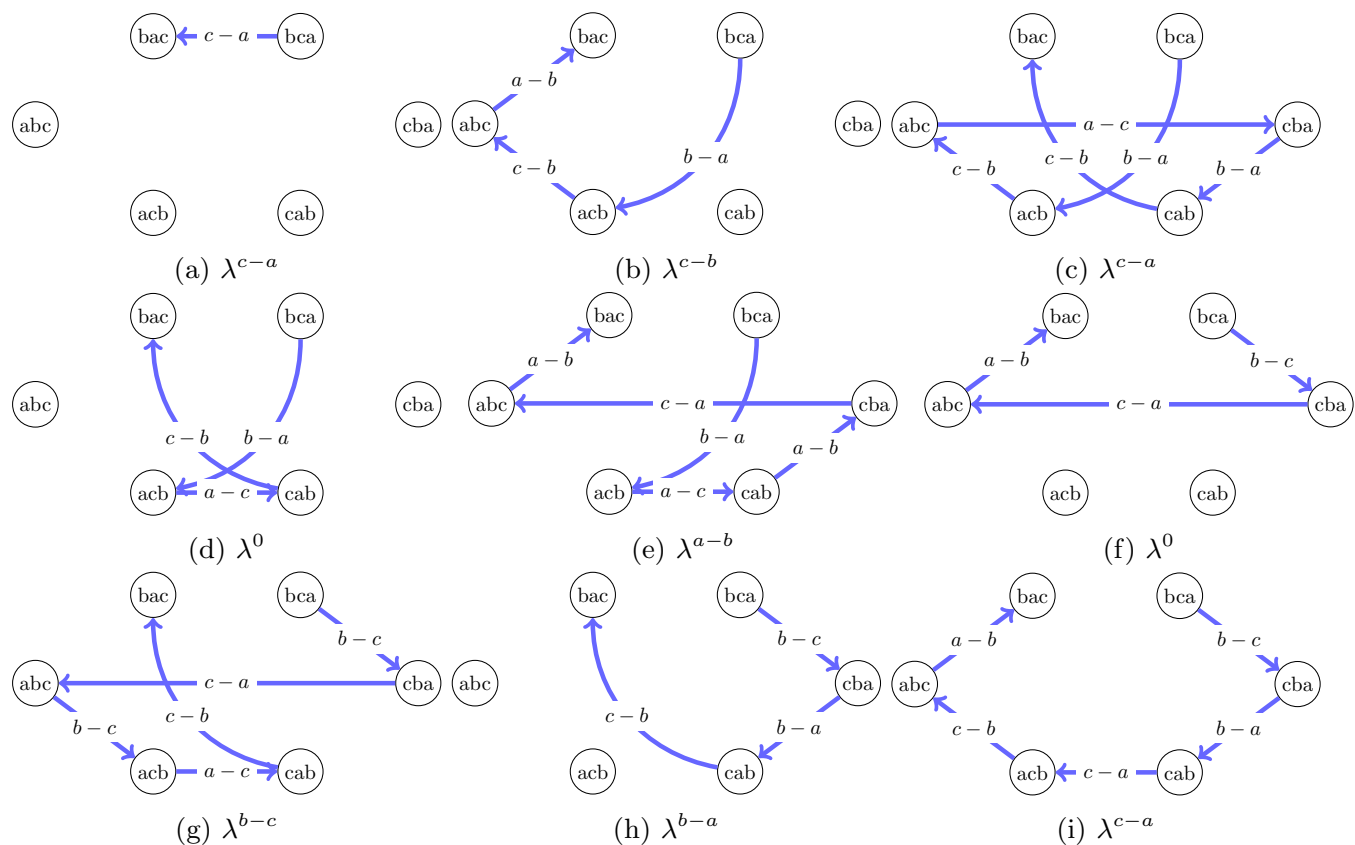


FIG. 8. All paths from (bca) to (bac). There are no bad trees for these two states.

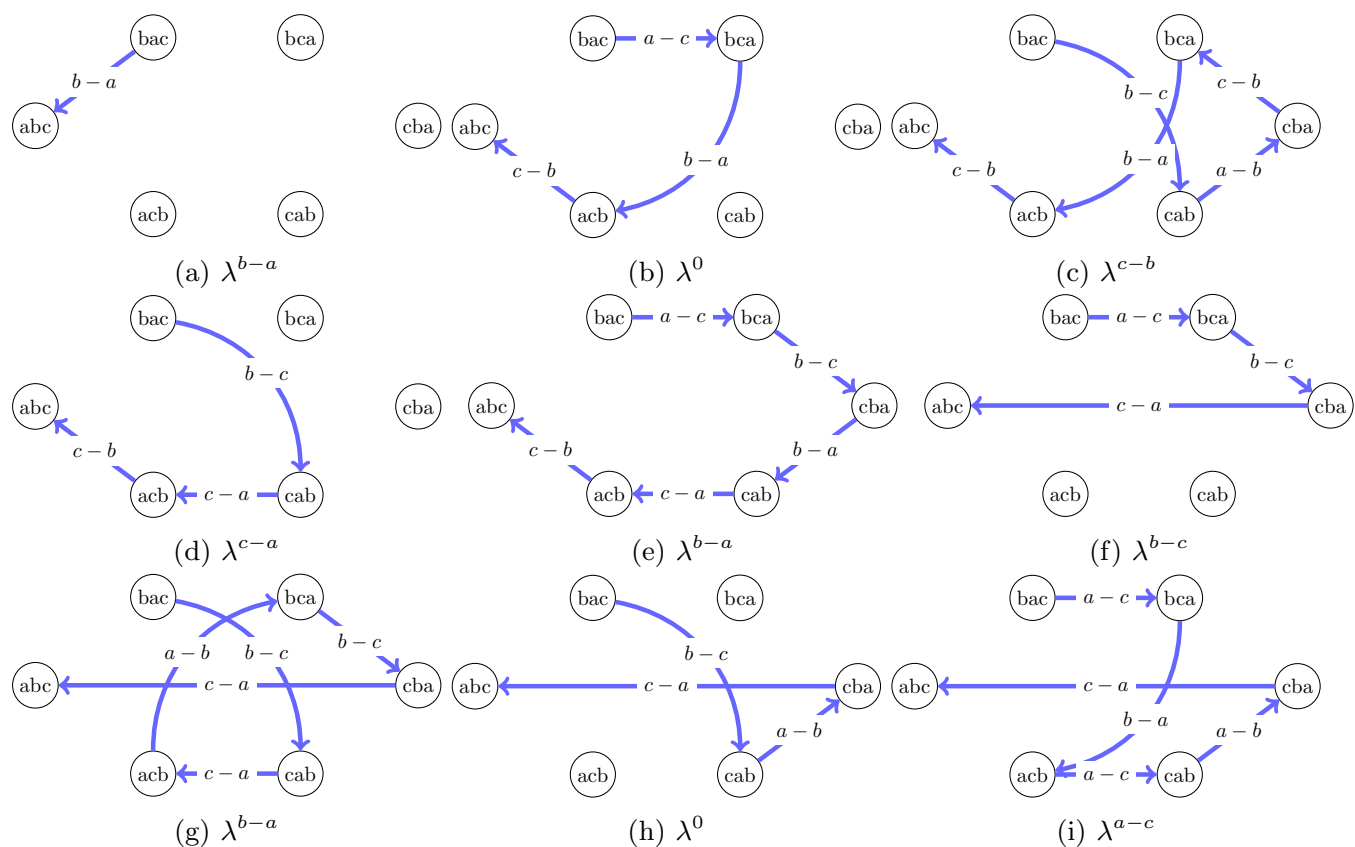


FIG. 9. All paths from (bac) to (abc). Bad trees include path (c) or (d).

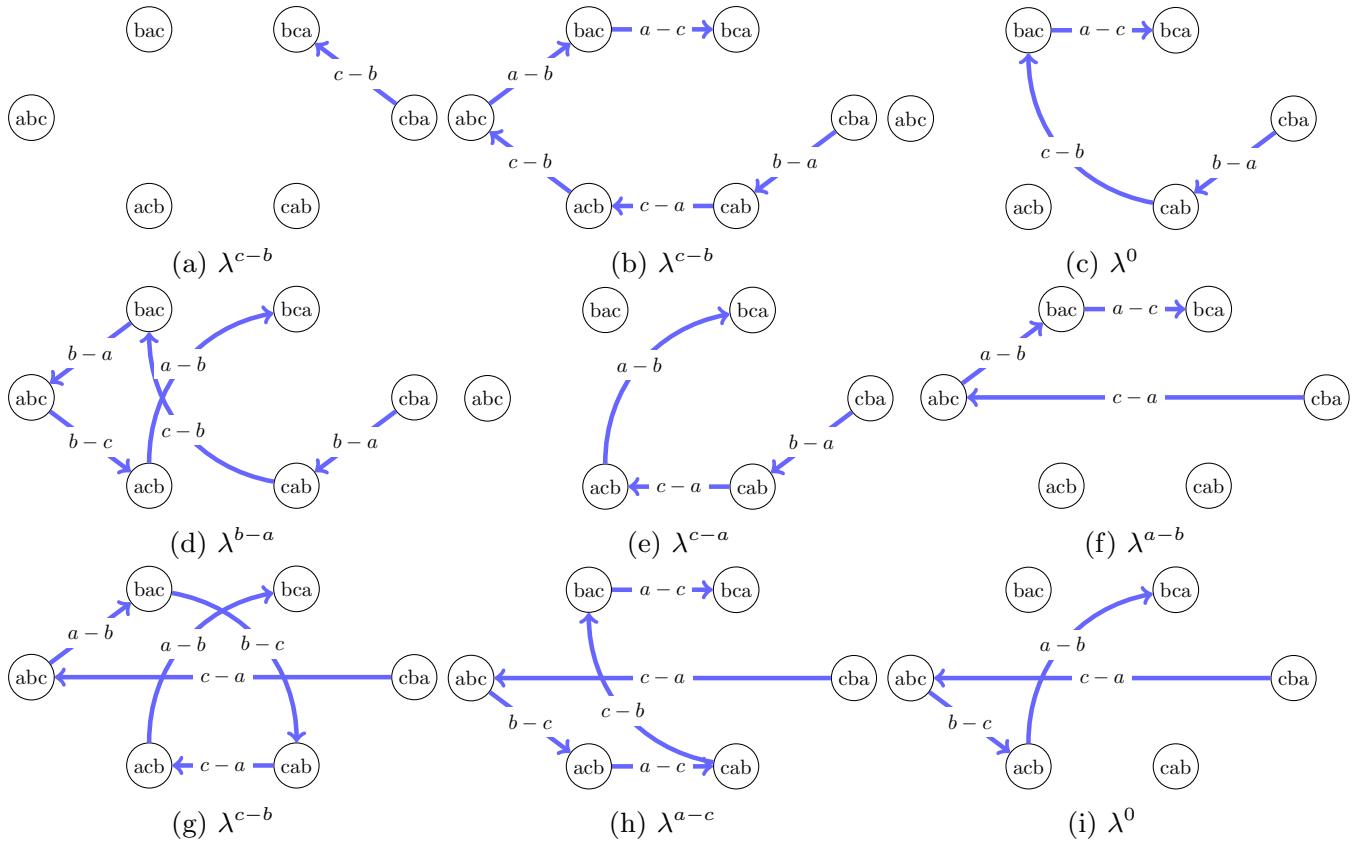


FIG. 10. All paths from (cba) to (bca). Bad trees include path (d) or (e).

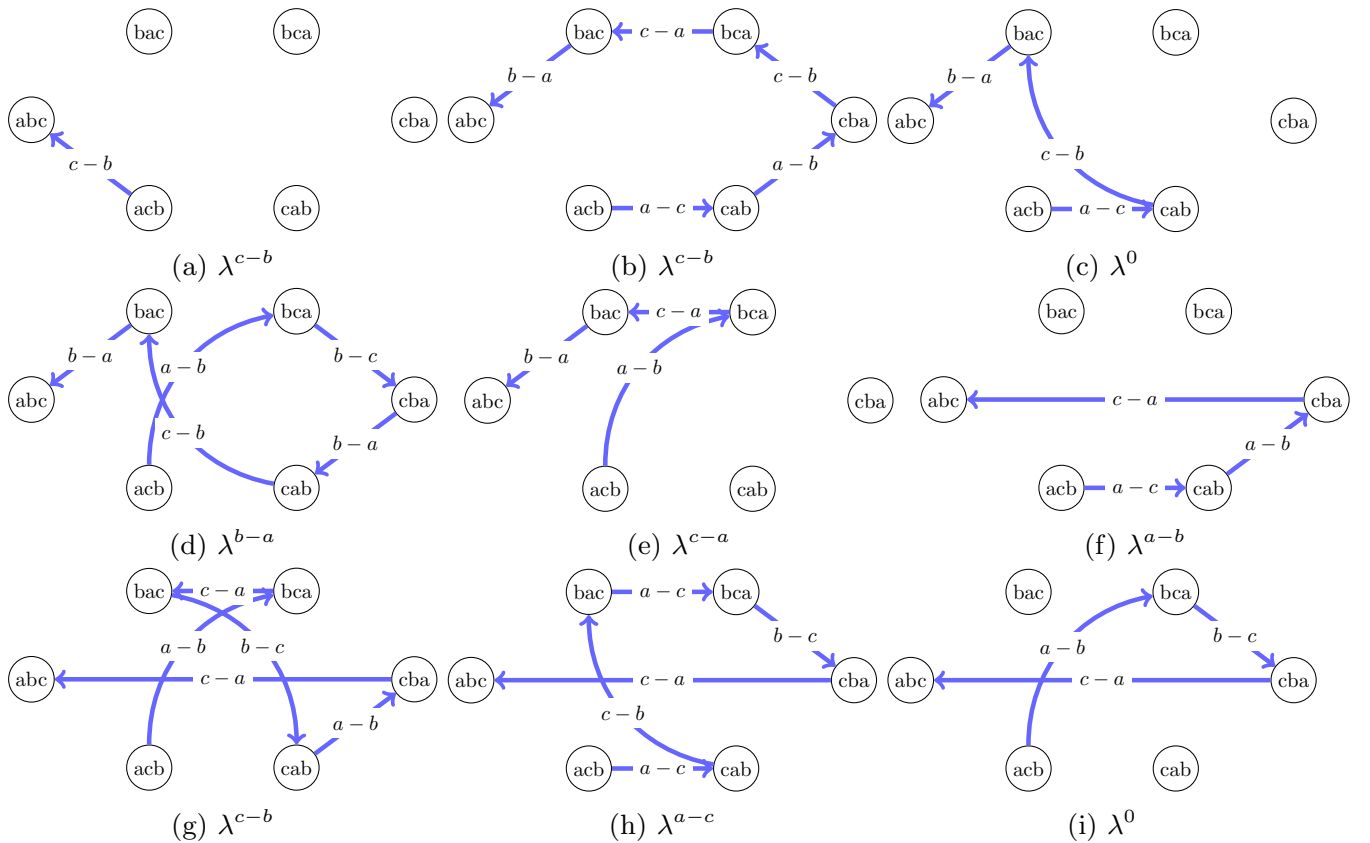


FIG. 11. All paths from (acb) to (abc). Bad trees include path (d) or (e).

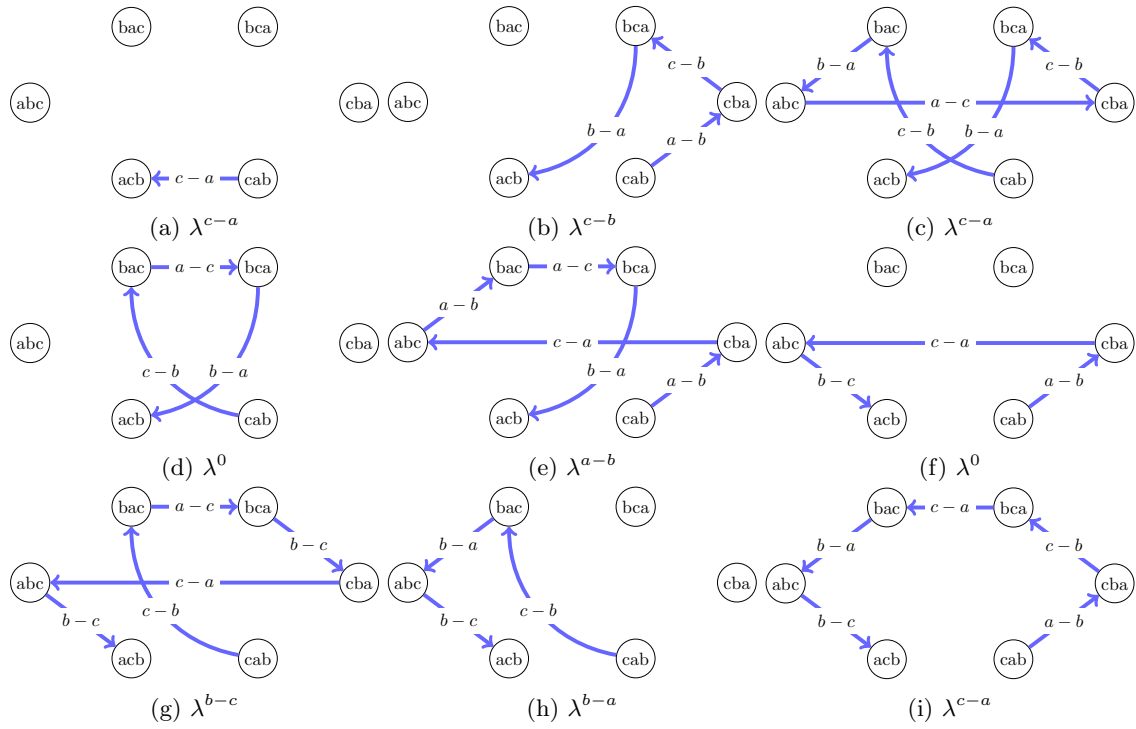


FIG. 12. All paths from (cab) to (acb). There are no bad trees for these two states.

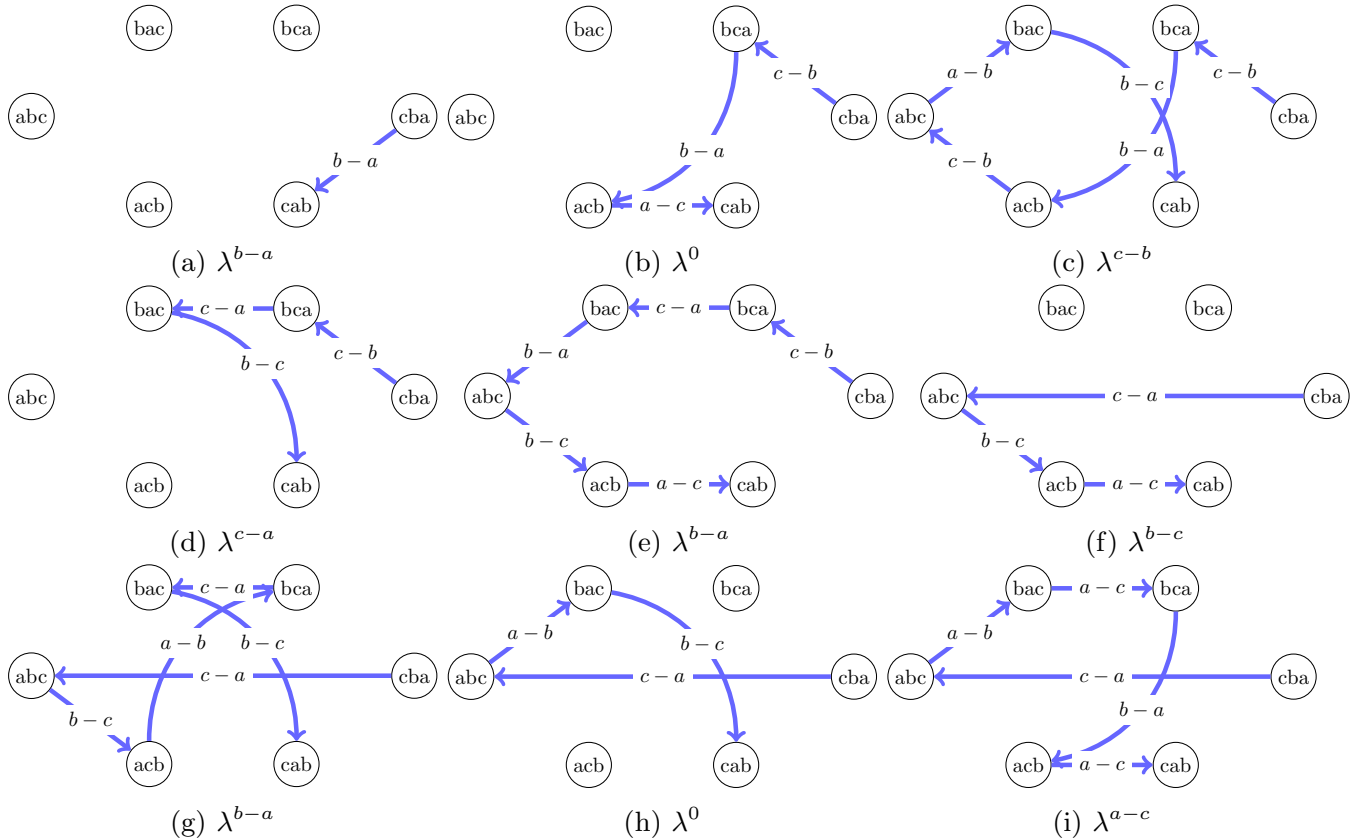


FIG. 13. All paths from (cba) to (cab). Bad trees include path (c) or (d).

of good trees with Figs. 9(b), 9(f), 9(h), and 9(i). Then,

$$\frac{\sum_{T \in \mathcal{BAD}(s)} \mathbf{P}(T) + \sum_{T \in \mathcal{GOOD}(s)} \mathbf{P}(T)}{\sum_{T \in \mathcal{BAD}(s)} \mathbf{P}(T') + \sum_{T \in \mathcal{GOOD}(s)} \mathbf{P}(T')} \leq \lambda^{2(b-a)}. \quad (\text{B3})$$

Proof. Let us use $x := b - a$, $y := c - b$, and $x + y := c - a$. The trees and their probabilities are shown in Figs. 14–18. We will show

$$\left[\sum_{T \in \mathcal{BAD}(s)} \mathbf{P}(T) + \sum_{T \in \mathcal{GOOD}(s)} \mathbf{P}(T) \right] \leq \lambda^{2x} \left[\sum_{T \in \mathcal{BAD}(s)} \mathbf{P}(T') + \sum_{T \in \mathcal{GOOD}(s)} \mathbf{P}(T') \right],$$

which obviously implies Eq. (B3). To get rid of the fractions in the probabilities we multiply them by the least common multiple of their denominators, i.e.,

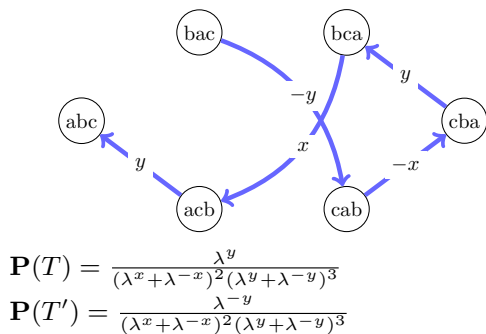
$$\text{LCM} := (\lambda^{x+y} + \lambda^{-x-y})^3 (\lambda^y + \lambda^{-y})^3 (\lambda^x + \lambda^{-x})^2.$$

We get for the left-hand side of the inequality

$$\begin{aligned} \text{LCM} & \left[\sum_{T \in \mathcal{BAD}(s)} \mathbf{P}(T) + \sum_{T \in \mathcal{GOOD}(s)} \mathbf{P}(T) \right] \\ &= 6\lambda^{5x+4y} + 16\lambda^{3x+4y} + 10\lambda^{x+4y} + 6\lambda^{5x+2y} + 26\lambda^{3x+2y} \\ &+ 42\lambda^{x+2y} + 18\lambda^{-x+2y} + 8\lambda^{x-2y} + 24\lambda^{-x-2y} \\ &+ 12\lambda^{-3x-2y} + 4\lambda^{-x-4y} + 4\lambda^{-3x-4y} + 10\lambda^{3x} + 36\lambda^x \\ &+ 42\lambda^{-x} + 8\lambda^{-3x}. \end{aligned}$$

And we get for the right-hand side

$$\begin{aligned} \lambda^{2x} \text{LCM} & \left[\sum_{T \in \mathcal{BAD}(s)} \mathbf{P}(T') + \sum_{T \in \mathcal{GOOD}(s)} \mathbf{P}(T') \right] \\ &= 10\lambda^{5x+4y} + 16\lambda^{3x+4y} + 6\lambda^{x+4y} + 18\lambda^{5x+2y} + 42\lambda^{3x+2y} \\ &+ 26\lambda^{x+2y} + 6\lambda^{-x+2y} + 8\lambda^{5x} + 42\lambda^{3x} + 36\lambda^x + 10\lambda^{-x} \\ &+ 12\lambda^{3x-2y} + 24\lambda^{x-2y} + 8\lambda^{-x-2y} + 4\lambda^{x-4y} + 4\lambda^{-x-4y}. \end{aligned}$$



The difference between the left- and the right-hand side is

$$\begin{aligned} & 4\lambda^{5x+4y} - 4\lambda^{x+4y} + 12\lambda^{5x+2y} - 12\lambda^{-x+2y} + 16\lambda^{3x+2y} \\ & - 16\lambda^{x+2y} + 8\lambda^{5x} - 8\lambda^{-3x} + 32\lambda^{3x} - 32\lambda^{-x} \\ & + 12\lambda^{3x-2y} - 12\lambda^{-3x-2y} + 16\lambda^{x-2y} - 16\lambda^{-x-2y} + 4\lambda^{x-4y} \\ & - 4\lambda^{-3x-4y}, \end{aligned}$$

and we can pair up the terms to conclude that this difference is positive. ■

The (bca) versus (cba) case. Note that $\frac{\pi_{\text{adj}}(\text{bca})}{\pi_{\text{adj}}(\text{cba})} = \lambda^{2(c-b)}$.

Figure 10 shows all paths from $s' = (\text{cba})$ to $s = (\text{bca})$. We can combine the bad trees with s' -path Figs. 10(d) or 10(e), and the good trees with paths Figs. 10(c), 10(f), 10(h) and 10(i) to show that the ratio between all trees and their reversals is smaller than $\lambda^{2(c-b)}$.

Lemma 25. For the bad trees $\mathcal{BAD}(s) \subset \mathcal{T}(a)$ with s' -path Figs. 10(d) or 10(e), and the good trees $\mathcal{GOOD}(s) \subset \mathcal{T}(a)$ with paths Figs. 10(c), 10(f), 10(h), and 10(i) it holds that

$$\frac{\sum_{T \in \mathcal{BAD}(s)} \mathbf{P}(T) + \sum_{T \in \mathcal{GOOD}(s)} \mathbf{P}(T)}{\sum_{T \in \mathcal{BAD}(s)} \mathbf{P}(T') + \sum_{T \in \mathcal{GOOD}(s)} \mathbf{P}(T')} \leq \lambda^{2(c-b)}.$$

Proof. We proceed as in the proof for Lemma 24 and finally get the difference between the left and the right-hand side:

$$\begin{aligned} & 4\lambda^{4x+5y} - 4\lambda^{4x+y} + 12\lambda^{2x+5y} - 12\lambda^{2x-y} + 16\lambda^{2x+3y} \\ & - 16\lambda^{2x+y} + 8\lambda^{5y} - 8\lambda^{-3y} + 32\lambda^{3y} - 32\lambda^{-y} + 12\lambda^{-2x+3y} \\ & - 12\lambda^{-2x-3y} + 16\lambda^{-2x+y} - 16\lambda^{-2x-y} + 4\lambda^{-4x+y} \\ & - 4\lambda^{-4x-3y}. \end{aligned}$$

The other cases. There are three other pairs of states for which we need to prove Lemma 21, namely the (acb) versus (abc) case, the (cab) versus (acb) case, and the (cba) versus (cab) case. However, the procedure is always the same: We identify the bad and good trees and combine them to conclude Eq. (11). For the missing paths consider Figs. 11–13. Moreover, it turns out that these missing cases are somehow symmetric to the previous cases. Indeed, the paths from (acb) to (abc) in Fig. 11 include an equivalent set of weights of edges as those from (cba) to (bca) in Fig. 10. The same is true for the cases (bca) versus (bac) and (cab) versus (acb) (see Figs. 8 and 12), and the cases (bac) versus (abc) and (cba) versus (cab) (see Figs. 9 and 13).

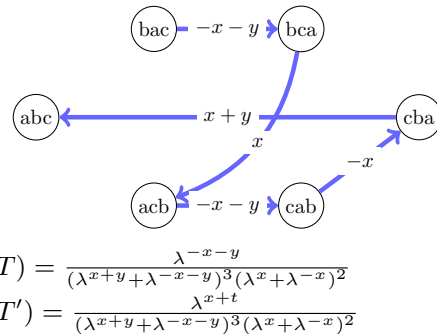
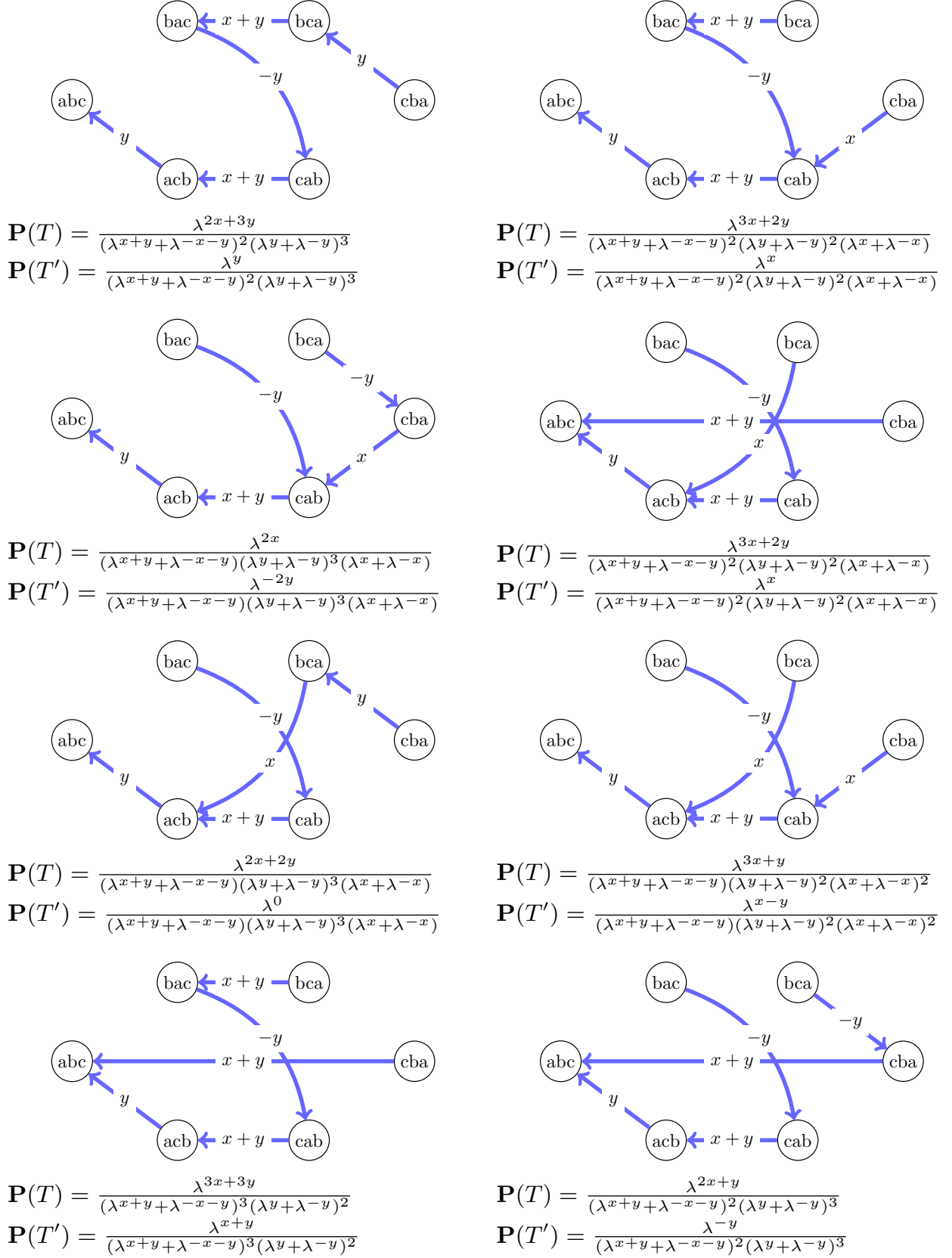
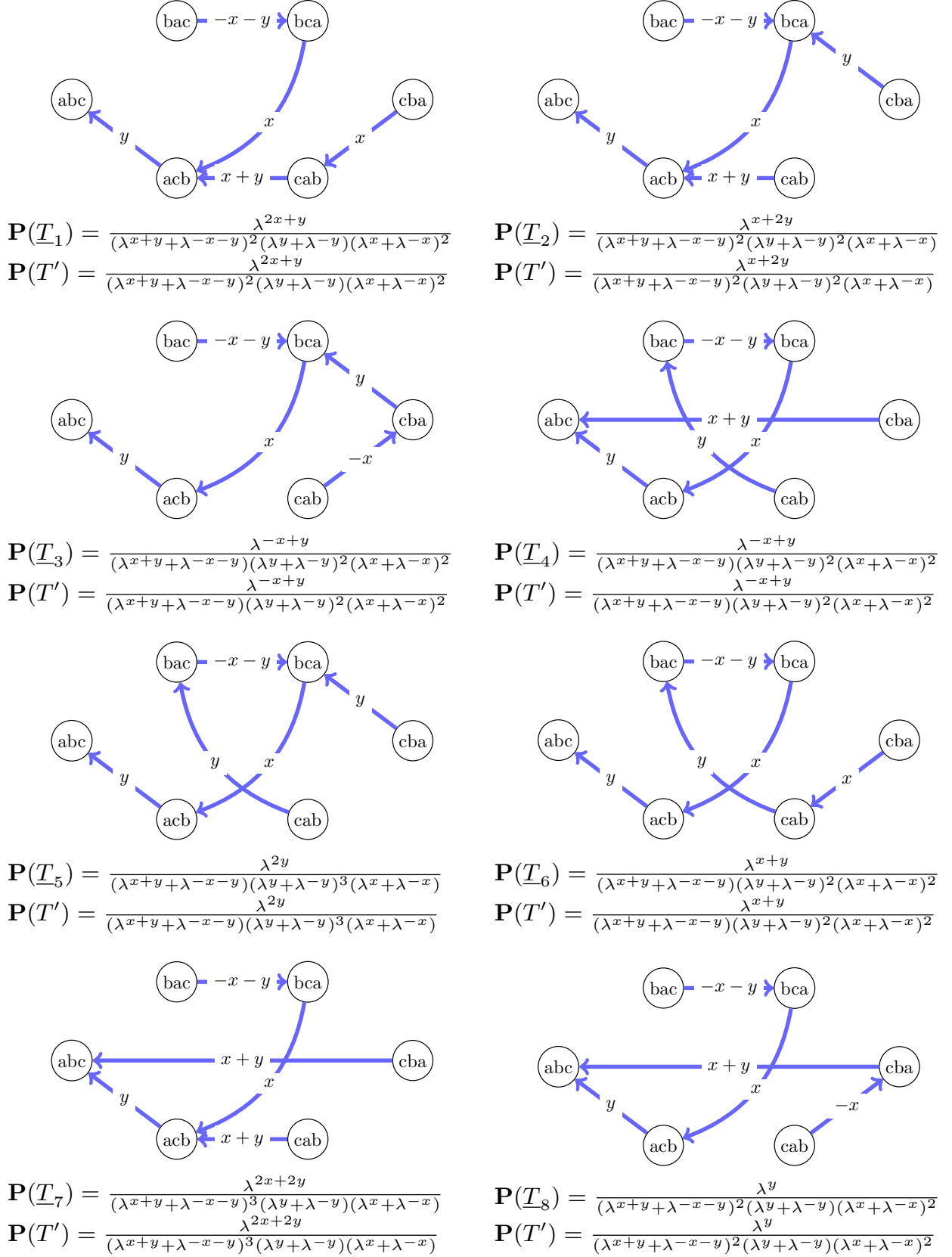
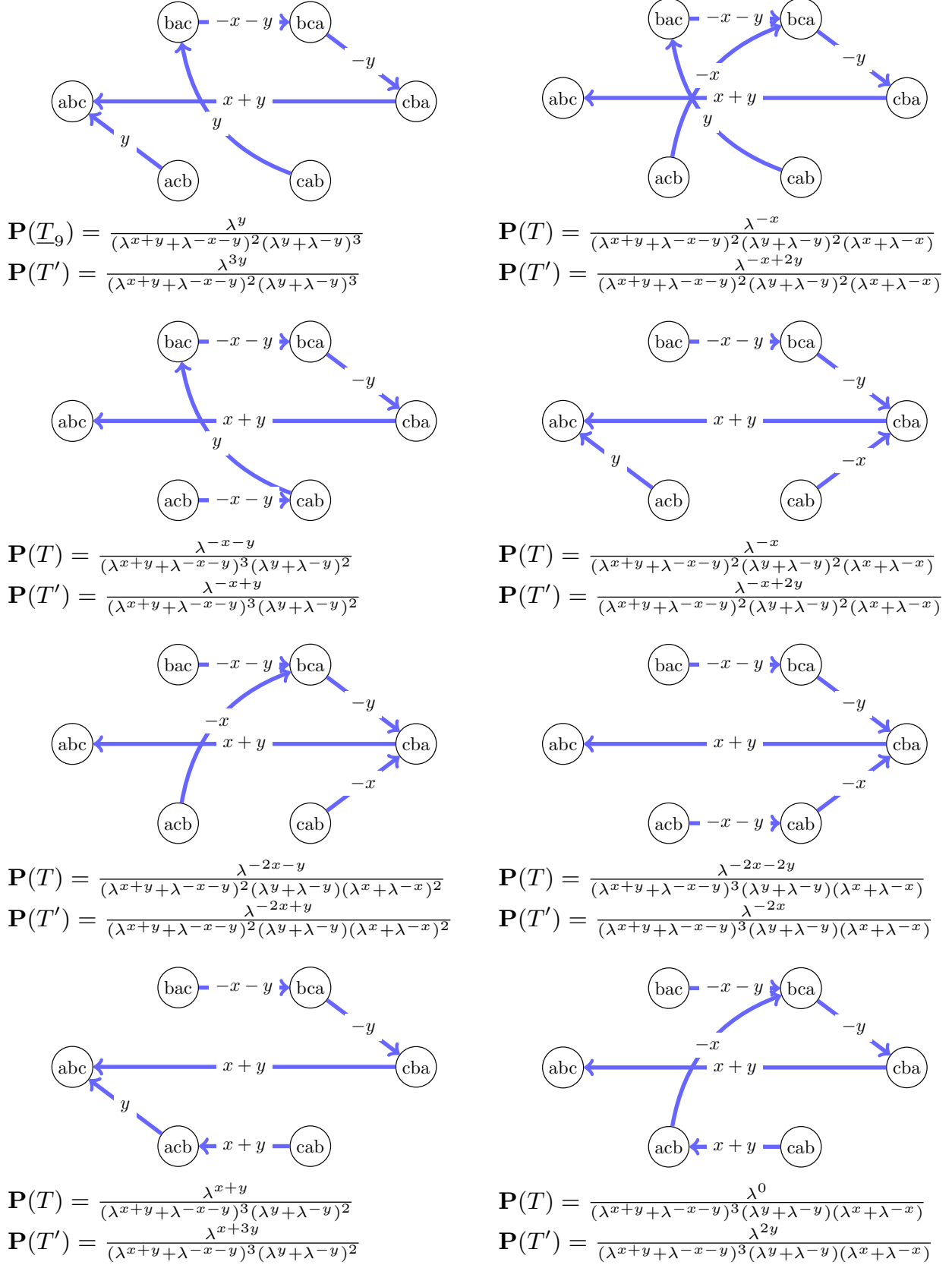


FIG. 14. Bad tree with path 9(c): (bac) → (cab) → (cba) → (bca) → (acb) → (abc), and good tree with path 9(i): (bac) → (bca) → (acb) → (cab) → (cba) → (abc).

FIG. 15. Bad trees with path 9(d): $(bac) \rightarrow (cab) \rightarrow (acb) \rightarrow (abc)$.

FIG. 16. Good trees with path 9(b): $(bac) \rightarrow (bca) \rightarrow (acb) \rightarrow (abc)$.

FIG. 17. Good trees with path 9(f): $(bac) \rightarrow (bca) \rightarrow (cba) \rightarrow (abc)$.

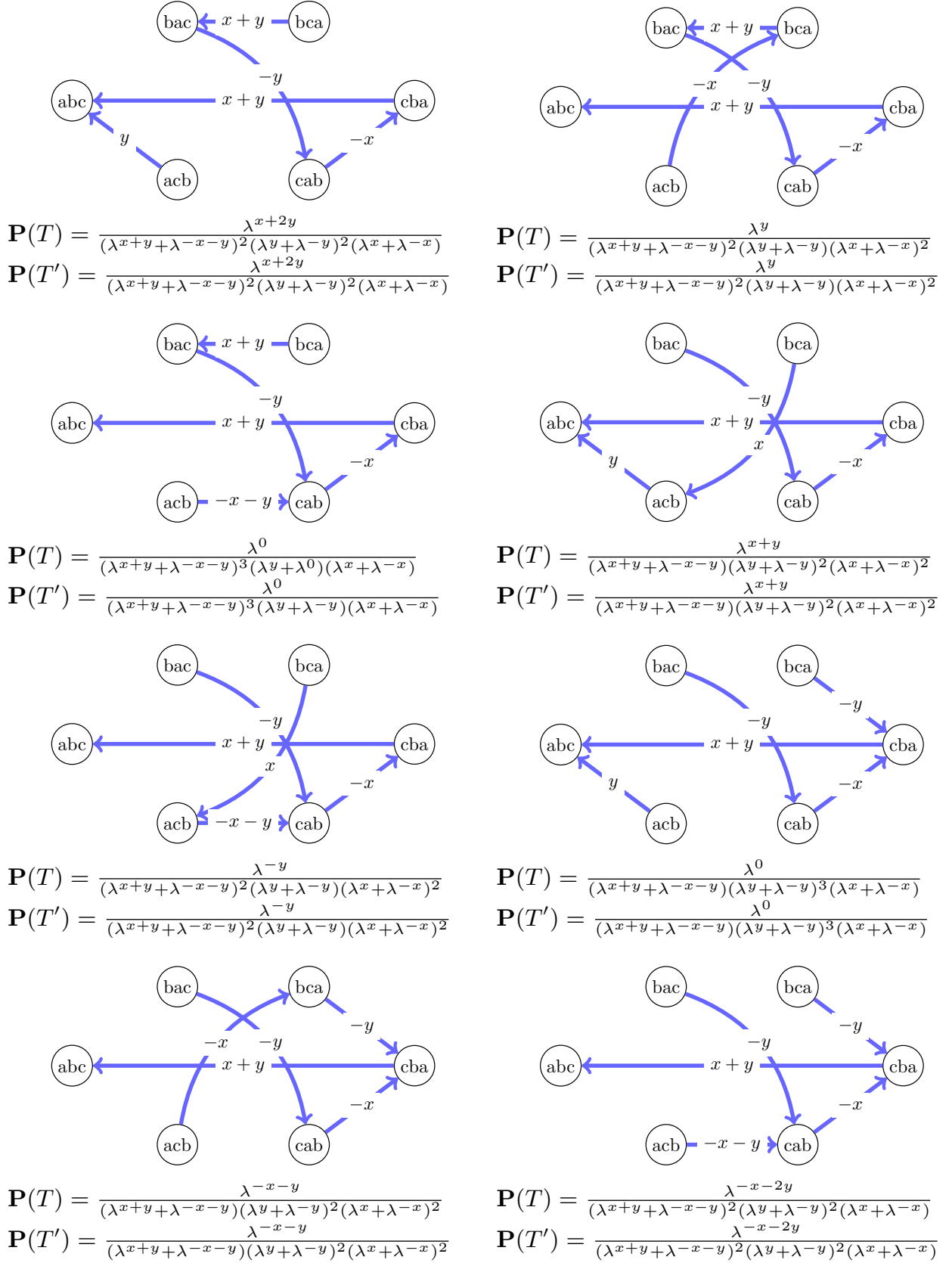


FIG. 18. Good trees with path 9(h): (bac) → (cab) → (cba) → (abc).

- [1] B. Geissmann and P. Penna, Sort well with energy-constrained comparisons, *CoRR*, abs/1610.09223 (2016).
- [2] M. Ajtai, V. Feldman, A. Hassidim, and J. Nelson, Sorting and selection with imprecise comparisons, *ACM Trans. Algor.* **12**, 19 (2016).
- [3] L. E. Blume, The statistical mechanics of strategic interaction, *Games Econ. Behav.* **5**, 387 (1993).
- [4] L. L. Thurstone, A law of comparative judgment, *Psychol. Rev.* **34**, 273 (1927).
- [5] K. Palem and A. Lingamneni, Ten years of building broken chips: The physics and engineering of inexact computing, *ACM Trans. Embedded Comput. Syst.* **12**, 87 (2013).
- [6] L. Alonso, P. Chassaing, F. Gillet, S. Janson, E. M. Reingold, and R. Schott, Quicksort with unreliable comparisons: A probabilistic analysis, *Combinator. Probabil. Comput.* **13**, 419 (2004).
- [7] M. Braverman and E. Mossel, Noisy sorting without resampling, in *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (SIAM, Philadelphia, PA, USA, 2008), pp. 268–276.
- [8] P. Hadjicostas and K. B. Lakshmanan, Recursive merge sort with erroneous comparisons, *Discrete Appl. Math.* **159**, 1398 (2011).
- [9] M. Mezard and A. Montanari, *Information, Physics, and Computation* (Oxford University Press, Oxford, 2009).
- [10] T. M. Liggett, *Interacting Particle Systems*, Vol. 276 (Springer Science & Business Media, Berlin, 2012).
- [11] H. Spohn, *Large Scale Dynamics of Interacting Particles* (Springer Science & Business Media, Berlin, 2012).
- [12] C. T. MacDonald, J. H. Gibbs, and A. C. Pipkin, Kinetics of biopolymerization on nucleic acid templates, *Biopolymers* **6**, 1 (1968).
- [13] C. A. Tracy and H. Widom, Asymptotics in ASEP with step initial condition, *Commun. Math. Phys.* **290**, 129 (2009).
- [14] B. Derrida, An exactly soluble non-equilibrium system: The asymmetric simple exclusion process, *Phys. Rep.* **301**, 65 (1998).
- [15] S. A. Janowsky and J. L. Lebowitz, Finite-size effects and shock fluctuations in the asymmetric simple-exclusion process, *Phys. Rev. A* **45**, 618 (1992).
- [16] F. Spitzer, *Interaction of Markov Processes* (Birkhäuser, Boston, MA, 1991), pp. 66–110.
- [17] G. de With, *Appendix C: The Lattice Gas Model* (Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Germany, 2013), pp. 487–494.
- [18] C. N. Yang and T. D. Lee, Statistical theory of equations of state and phase transitions. I. Theory of condensation, *Phys. Rev.* **87**, 404 (1952).
- [19] K. Nagel and M. Schreckenberg, A cellular automaton model for freeway traffic, *J. Phys. I France* **2**, 2221 (1992).
- [20] C. Godrèche, *Solids Far from Equilibrium* (Cambridge University Press, Cambridge, UK, 1992).
- [21] D. Kandel and E. Domany, Rigorous derivation of domain growth kinetics without conservation laws, *J. Stat. Phys.* **58**, 685 (1990).
- [22] D. Kandel, E. Domany, and B. Nienhuis, A six-vertex model as a diffusion problem: Derivation of correlation functions, *J. Phys. A* **23**, L755 (1990).
- [23] C. L. Mallows, Non-null ranking models. I, *Biometrika* **44**, 114 (1957).
- [24] R. A. Bradley and M. E. Terry, Rank analysis of incomplete block designs: I. The method of paired comparisons, *Biometrika* **39**, 324 (1952).
- [25] D. B. Wilson, Mixing times of lozenge tiling and card shuffling Markov chains, *Ann. Appl. Probabil.* **14**, 274 (2004).
- [26] R. Bubley and M. Dyer, Faster random generation of linear extensions, *Discrete Math.* **201**, 81 (1999).
- [27] I. Benjamini, N. Berger, C. Hoffman, and E. Mossel, Mixing times of the biased card shuffling and the asymmetric exclusion process, *Trans. Amer. Math. Soc.* **357**, 3013 (2005).
- [28] P. Bhakta, S. Miracle, D. Randall, and A. P. Streib, Mixing times of Markov chains for self-organizing lists and biased permutations, in *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (SIAM, Philadelphia, PA, USA, 2013), pp. 1–15.
- [29] S. Haddadan and P. Winkler, Mixing of permutations by biased transposition, in *Proceedings of the 34th Symposium on Theoretical Aspects of Computer Science (STACS)*, Vol. 66 of LIPIcs (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017), pp. 41:1–41:13.
- [30] T. Gavenčiak, B. Geissmann, and J. Lengler, Sorting by swaps with noisy comparisons, in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (ACM, New York, NY, USA, 2017), pp. 1375–1382.
- [31] P. Diaconis and A. Ram, Analysis of systematic scan Metropolis algorithms using Iwahori-Hecke algebra techniques, *Michigan Math. J.* **48**, 157 (2000).
- [32] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times* (American Mathematical Society, Providence, RI, 2009).
- [33] F. P. Kelly, *Reversibility and Stochastic Networks* (Cambridge University Press, Cambridge, 2011).
- [34] M. Freidlin and A. D. Wentzell, *Random Perturbations of Dynamical Systems* (Springer Verlag, New York, 1984).
- [35] S. Greenberg, A. Pascoe, and D. Randall, Sampling biased lattice configurations using exponential metrics, in *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (SIAM, Philadelphia, PA, USA, 2009), pp. 76–85.
- [36] M. E. Dyer and C. S. Greenhill, A more rapidly mixing Markov chain for graph colorings, *Random Struct. Algorithms* **13**, 285 (1998).
- [37] P. Diaconis, S. Holmes, and R. M. Neal, Analysis of a nonreversible Markov chain sampler, *Ann. Appl. Probabil.* **10**, 726 (2000).
- [38] F. Chen, L. Lovász, and I. Pak, Lifting Markov chains to speed up mixing, in *Proceedings of the 31st annual ACM Symposium on Theory of Computing (STOC)* (ACM, New York, NY, USA, 1999), pp. 275–281.
- [39] R. Montenegro and P. Tetali, Mathematical aspects of mixing times in Markov chains, *Found. Trends Theoret. Comput. Sci.* **1**, 237 (2006).
- [40] S. N. Evans, B. Sturmfels, and C. Uhler, Commuting birth-and-death processes, *Ann. Appl. Probabil.* **20**, 238 (2010).