Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Transfer matrix algorithm for computing the exact partition function of a square lattice polymer

Julian Lee

Department of Bioinformatics and Life Science, Soongsil University, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 2 May 2017
Received in revised form 17 March 2018
Accepted 25 March 2018
Available online 9 April 2018

Keywords:
Lattice polymer
Transfer matrix
Exact enumeration
Interacting self-avoiding walk

ABSTRACT

I develop a transfer matrix algorithm for computing the exact partition function of a square lattice polymer with nearest-neighbour interactions by extending a previous algorithm for computing the total number of self-avoiding walks. The computation time scales as $\sim 1.6^N$ with the chain length N, in contrast to the explicit enumeration where the scaling is $\sim 2.7^N$. The exact partition function can be obtained faster with the transfer matrix method than with the explicit enumeration for $N > 25$. The new results for up to $N = 42$ are presented.

1. Introduction

Polymers play important roles in various fields of science, including biology, where various biopolymers perform crucial functions for life processes. Although the properties of heteropolymers such as proteins are most interesting, many important general properties of polymers can be learned from simpler homopolymer models. The simplest toy models for studying such a polymer are lattice models, such as two-dimensional square or three-dimensional cubic lattice polymers [1–4]. By introducing hydrophobic inter-monomer interactions, a lattice model can be used as a model for a polymer in a dilute solution [1–30]. Various quantities such as the radius of gyration, end-to-end distance, and specific heat have been calculated for the lattice models.

One important advantage of the lattice polymer is that all the possible conformations can be enumerated exactly [29–32]. The exact partition function for lattice polymers up to $N = 28$ for cubic lattices and $N = 40$ for square lattices has been computed by a recently developed efficient enumeration algorithm [32], where N is the number of monomers in the polymer. The most serious obstacle for the explicit enumeration of lattice polymer conformations of longer chain lengths is the fact that the number of conformations and the corresponding computational time grows exponentially with the chain length, as $\sim 2.7^N$ [31,32].

In this work, I propose a new transfer matrix approach where the exact partition function of a square lattice polymer can be computed much faster than using the explicit enumeration for long chains. In the transfer matrix method, instead of generating one conformation at a time, one keeps track of an ensemble of partially built conformations. By discarding detailed information on the partially built conformations and retaining only the essential information required for the computation of the partition function, the transfer matrix method drastically reduces the computational time without sacrificing the exact nature of the computation. The transfer matrix approach has been mostly used for computing the partition function for spin systems [33,34], including a simple model of proteins [35,36]. The most relevant previous work is the transfer matrix method used for the enumeration of self-avoiding walks (SAWs) on the square lattice [37]. This method is an improvement of earlier methods for enumerating SAWs [38,39], and also an extension of the methods that enumerates the self-avoiding polygons (SAPs) on the square lattice [40–42]. Because a conformation of a lattice polymer is equivalent to a SAW, the *total* number of polymer conformations on the square lattice is enumerated by this method. The computation has been performed for up to $N = 80$ [37]. We generalize this method so that the nearest-neighbour contact between the monomers can be taken into account. By computing the number of conformations for each value of the contact number, the exact partition function can be computed as a function of the temperature. We find that the computational time scales as $\sim 1.6^N$, in contrast to $\sim 2.7^N$ of the explicit enumeration. The partition function can be obtained much faster with the transfer matrix method than with the explicit enumeration for $N > 25$. All the known results up to $N = 40$ can be reproduced within a day with a *single* CPU. The new results for $N = 41$ and $N = 42$ will also be presented.

E-mail address: jul@ssu.ac.kr.

https://doi.org/10.1016/j.cpc.2018.03.022
0010-4655/© 2018 Published by Elsevier B.V.

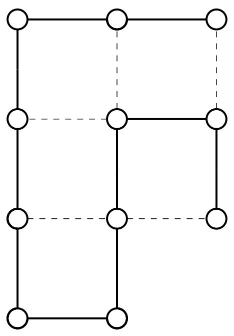

Fig. 1. Example of a conformation of a square lattice polymer, with $N = 11$ and $K = 5$. The non-bonded contacts are denoted by dashed lines.

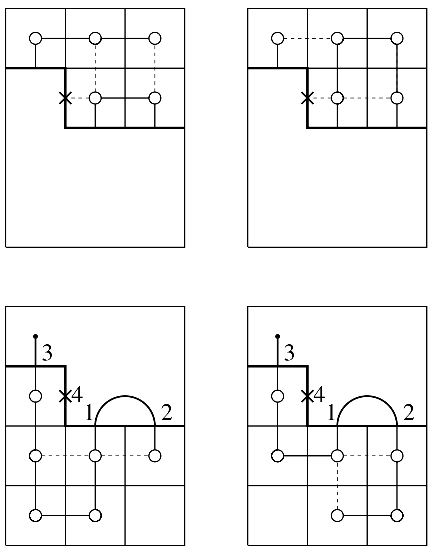

Fig. 2. Example of building polymer conformations of length $N = 11$ spanning the box of rectangle 3×4. The current cut-line is shown as the thick line. The cell at $(i, j) = (1, 3)$ has been just been completed, and the signature at the cut-line is (3 4 1 2). An empty edge with a cross on it denotes that the site just beneath the edge is occupied. Two examples of a partially built conformation with $n = 6$, $k = 2$, corresponding to this signature, are also shown below the cut-line at the bottom of the figure. Two examples of the upper parts of the conformations with $K = 5$, generated from this cut-line signature, are shown at the top.

2. Model

We consider a polymer on a square lattice, where a pair of non-bonded monomers that are neighbouring with each other is regarded as being in contact. The energy value of $-\epsilon$ is associated with each of these nearest-neighbour contacts. Therefore, the energy of a given conformation can be expressed as $E = -\epsilon K$, where K is the number of contacts formed in the conformation, which will simply be called the contact number from now on. An example of a square-lattice polymer conformation with $N = 11$ and $K = 5$ is shown in Fig. 1. The partition function is then given by

$$Z = \sum_{\text{all confs}} e^{-\beta E} = \sum_{K=0}^{K_{\max}} \Omega(K) e^{\beta\epsilon K}, \qquad (1)$$

where $\beta \equiv 1/k_B T$ and $\Omega(K)$ indicate the number of conformations for a given contact number K, also called the density of states. For the polymer on the square lattice, the maximum number of possible contacts $K_{\max}$ is given as [1]:

$$K_{\max}(N) = \begin{cases} N - 2m & \text{for } m^2 < N \le m(m+1), \\ N - 2m - 1 & \text{for } m(m+1) < N \le (m+1)^2, \end{cases} \qquad (2)$$

where m is a positive integer and N is the number of monomers forming the polymer chain. It is clear from Eq. (1) that the partition function for any temperature T can be computed once the density of states $\Omega(K)$ is obtained. The purpose of the algorithm developed in the current study is the efficient computation of $\Omega(K)$. The current model is also called the interacting self-avoiding walk (ISAW) on the square lattice. When $\beta = 0$, the partition function of ISAW in Eq. (1) gets reduced to the *total* number of SAWs that has been computed by the transfer matrix approach [37,38].

3. Transfer matrix method

The transfer matrix method developed in the current study is based on a previous method for enumerating the total number of SAWs on the square lattice [37]. First, the conformations are classified according to the rectangular box they span. For a given box, only the conformations touching all four walls of the box are enumerated. This idea has also been implemented in a parallel algorithm for explicit enumeration [31]. Let us denote the width and height of the box as w and h, respectively. It is convenient to visualize the box as consisting of $w \times h$ cells, with each cell enclosing a lattice site (Fig. 2). Because the conformation is required to touch all the walls of the box, we get the upper bound for the box size, $w + h - 1 \le N$. In fact, the number of conformations spanning the box with $w + h - 1 = N$ can be obtained with an analytic formula, so only the conformations spanning the boxes with $w + h - 1 < N$ need to be enumerated [31]. Because the polymer conformations must fit inside the box, there is also a lower bound $w \times h \ge N$.

In the transfer matrix method, a cut-line bisecting the lattice is considered, which is moved to build conformations, cell by cell (Fig. 2). The main idea of the transfer matrix method is to count the number of partial conformations built up to the cut-line, and use this information to obtain the number of partial conformations when the cut-line is moved so that the next cell is incorporated into the lattice space for the partial conformation. This iterative procedure eventually leads to the full density of states when the cut-line reaches the top of the box and all the cells are incorporated. We will take the convention that the initial cut-line is at the bottom of the box, which is moved upwards as the algorithm proceeds. For a given row, the cells will be constructed from left to right. Denoting the coordinates of a cell as (i, j) $(1 \le i \le w,\ 1 \le j \le h)$, the cut-line has a kink on the right-hand side of the cell that is included in the partial conformation most recently, and consequently there are $w + 1$ edges in the cut-line (Fig. 2).

In an earlier version of the algorithm for SAW, the partially built conformations were classified according to their topology of the connection to the current cut-line [38]. In the improved newer version, they were classified by the connection topology of the part that is yet to be built, leading to a much simpler procedure for pruning out unnecessary conformations [37]. This topological information can be represented by a sequence of digits s_i $(i = 1, \ldots, w + 1)$, called the cut-line signature, associated with each edge of the cut-line. In the algorithm for SAW, each digit ranges from 0 to 3, where 0 represents no line crossing the edge, called the empty edge, 1 and 2 the left and the right stems of a loop, respectively, and 3 the free end [37]. The new element in the transfer matrix method for ISAW is that in order to keep track of the contact numbers of partial conformations, we need to introduce two types of empty edges depending on whether the site just below the empty edge is occupied or not, denoted by the digits 0 and 4, respectively. Therefore, in the algorithm for ISAW, each digit of the signature ranges from 0 to 4 (Fig. 2). We will refer to the edge with the digit 4 as being "charged". At most two digits of the signature can take the value 3, and the number of digits with the values 1 and 2 must be equal [37,38].

The number of partial conformations $F(n, k, v, s)$, called the partial density of states, is recorded for given values of the chain length

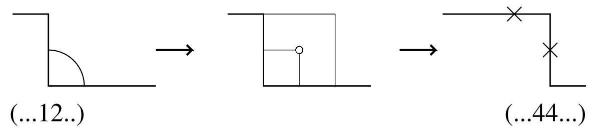

Fig. 3. Incoming pair of lines (1 2) forms a loop at the new cell to yield a pair of digits $(s_i, s_{i+1}) = (4, 4)$ in the new signature. The partial chain length n increases by one. The partial contact number k remains unchanged.

n, the contact number k of the partial conformations, a variable v that records whether the partial conformation has touched the left and the right walls of the box, and the cutline signature s. We will call n and k the partial chain length and the partial contact number, respectively, from now on. The variable v is required for the purpose of pruning out unnecessary partial conformations [37,38], as will be explained later. The value of v takes four possible values depending on whether the partial conformation has touched either of the two walls.

In the example of Fig. 2, polymer conformations of $N = 11$ spanning the rectangular box with $w = 3$ and $h = 4$ are considered. In the lower part of the figure, the conformations have been built up to the cell at $(i, j) = (1, 3)$, and the current cut-line is shown as the thick line. The signature, denoted as (3 4 1 2), describes how the lines coming out of this cut-line will eventually be connected: the line crossing the leftmost edge will become a free end, and the two lines crossing the rightmost edges will join each other to form a loop. The second edge is empty and also charged, because the site beneath the edge, inside the cell at (1, 3), is occupied by a monomer. Only the partial conformations consistent with these conditions are being considered. Two examples of such conformations that touch both of the left and right walls, with $n = 6$ and $k = 2$, are shown at the lower part of the figure. Full conformations generated from a signature must be consistent with the connection topology dictated by the signature. Two examples of the completion of the partial conformations, with $N = 11$ and $K = 5$, are shown at the upper part of the figure.

The cell-by-cell construction of an ensemble of partial conformations can be described in general terms by the update rules of a cell. This is exactly the same as that in the previous method for SAW [37], except that we distinguish empty edges according to whether the site directly beneath is occupied by a monomer, in order to keep track of contact numbers. At the moment, when a new cell is being added to the lattice space for the partial conformation, we will call the left and the bottom edges of the cell, which used to be the parts of the previous cut-line, the incoming edges. Similarly, the right and the top edges, which will become the parts of the new cut-line, will be called the outgoing edges. We will also call the polymer bonds that cross the incoming and outgoing edges the incoming and outgoing lines, respectively (Figs. 3–10).

The simplest case is when both of the incoming edges are occupied, as shown in Fig. 3. The lines crossing the left and bottom and edges must be 1 and 2 to be consistent. In this case, they join at the monomer in this cell and become part of the partial conformation below the new cut-line. The numbers $(s_i\ s_{i+1}) = (1\ 2)$ encoding this loop in the previous cut-line signature turn into charged empty edges $(s_i\ s_{i+1}) = (4\ 4)$ in the new signature. Any other incoming pair of lines leads to an inconsistency and is not allowed. In fact, the update rule for a single incoming line is such that incoming pairs of lines other than (1 2) never appear, as elaborated below. The partial chain length n increases by one and the partial contact number k remains unchanged after this update.

When there is a horizontal single incoming line of the form $(s_i\ s_{i+1}) = (A\ 0)$ or $(A\ 4)$, where $A = 1, 2,$ or 3, this line can continue through either the right or the top edge. The vertical and the horizontal continuation lead to the pair $(s_i\ s_{i+1}) = (A\ 4)$

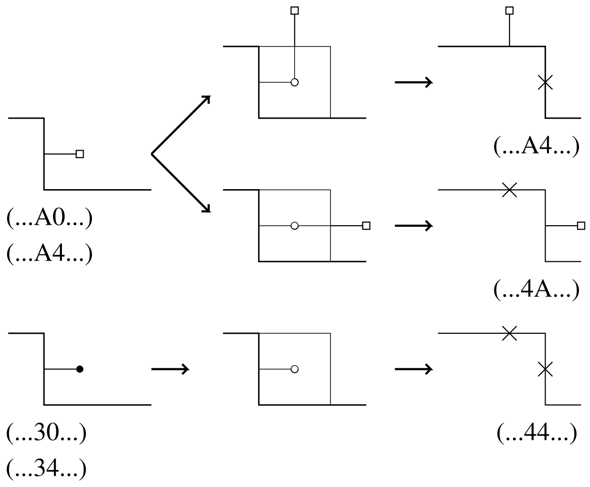

Fig. 4. Single incoming line from the left edge. The small square at the end of the line denotes either a loop stem or a free end, and the corresponding digit is denoted by $A = 1, 2,$ or 3. The incoming line can continue vertically or horizontally. A free end can also terminate at the new cell. The partial chain length n increases by one. The partial contact number k remains unchanged if the digit associated with the empty edge is 0, or increases by one if it is 4.

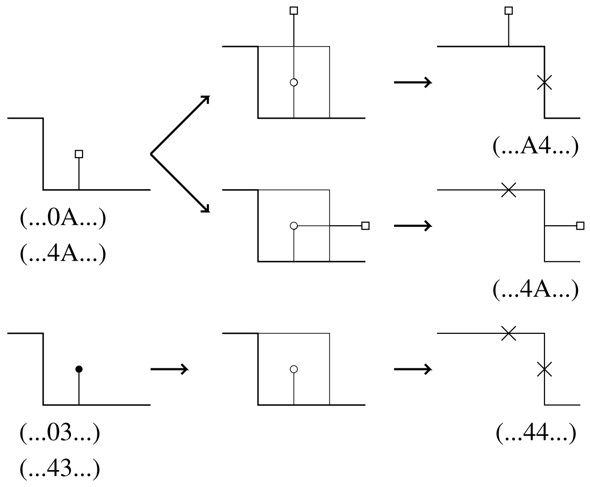

Fig. 5. Single incoming line from the bottom edge. A square denotes either a loop stem or a free end, and the corresponding digit is denoted by $A = 1, 2,$ or 3. The incoming line can continue vertically or horizontally. A free end can also terminate at the new cell. The partial chain length n increases by one. The partial contact number k remains unchanged if the digit associated with the empty edge is 0, or increases by one if it is 4.

and $(4\ A)$ in the new cut-line signature, respectively, as shown in the upper part of Fig. 4. The horizontal continuation is allowed only if $s_{i+2} = 0$, or $A = 1$ and $s_{i+2} = 2$. This ensures that the inconsistent pair of incoming lines at the next cell does not appear. For the case when the incoming line is a free end, $A = 3$, there is another possibility that the line terminates in the new cell. This leads to $(s_i\ s_{i+1}) = (4\ 4)$ in the new cut-line signature, as shown in the lower part of the figure. A similar update rule applies for the vertical input line (Fig. 5). Again, the horizontal continuation is allowed only if $s_{i+2} = 0$, or $A = 1$ and $s_{i+2} = 2$. The partial chain length n increases by one for all of these updates. The partial contact number k increases by one or remains unchanged depending on whether the empty edge is charged or not.

When there are no incoming lines, the simplest case is the one where there is no monomer in the new cell, with the resulting pair of digits given as $(s_i\ s_{i+1}) = (0\ 0)$ in the new signature (Fig. 6).

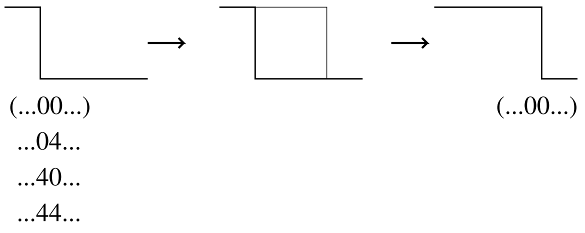

Fig. 6. No incoming lines. If there is no monomer in the cell, the resulting pair of digits is $(s_i, s_{i+1}) = (0, 0)$ in the new cut-line signature. The partial chain length n and the partial contact number k remain unchanged.

This is the only update where the partial chain length n remains the same as in the previous state. Consequently, k also remains unchanged.

The case with a monomer in the new cell is more complicated. This could be the first time we encounter a monomer of the partial conformation, in which case there may be one line or two lines connected to this monomer going vertically upwards or horizontally right, leading to one free end or two free ends (Fig. 7). Because we are considering a single connected chain, and the conformation is required to touch the bottom wall of the box, this kind of update is allowed only at the first row, and only for $n = 0$. As a consequence of $n = 0$, $s_k = 0$ for all k in the previous signature. The partial chain length and the partial contact number are $n = 1$ and $k = 0$ after any of these updates.

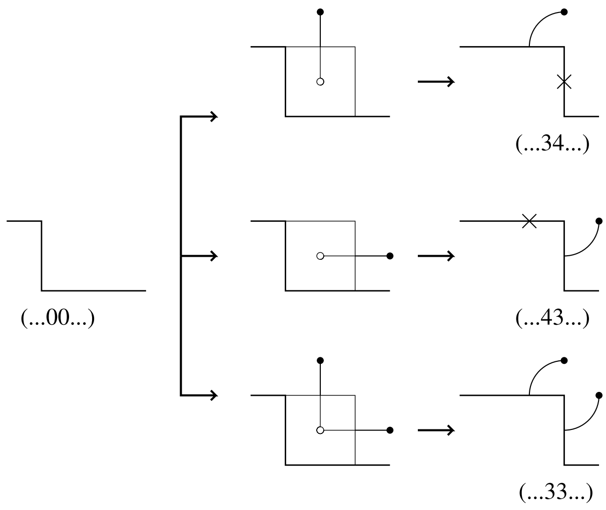

Fig. 7. No incoming lines, where all the digits of the previous signature are zero. The monomer in the cell is the first one to be encountered. The resulting pair of digits is $(s_i, s_{i+1}) = (3, 4), (4, 3)$, or $(3, 3)$, depending on the number of lines emitted from the monomer and the edge the line is crossing. The partial chain length n is one and the partial contact number k is zero after any of these updates.

If the monomer in the current cell is not the first monomer we encountered, then it is a part of the loop or free end protruding out

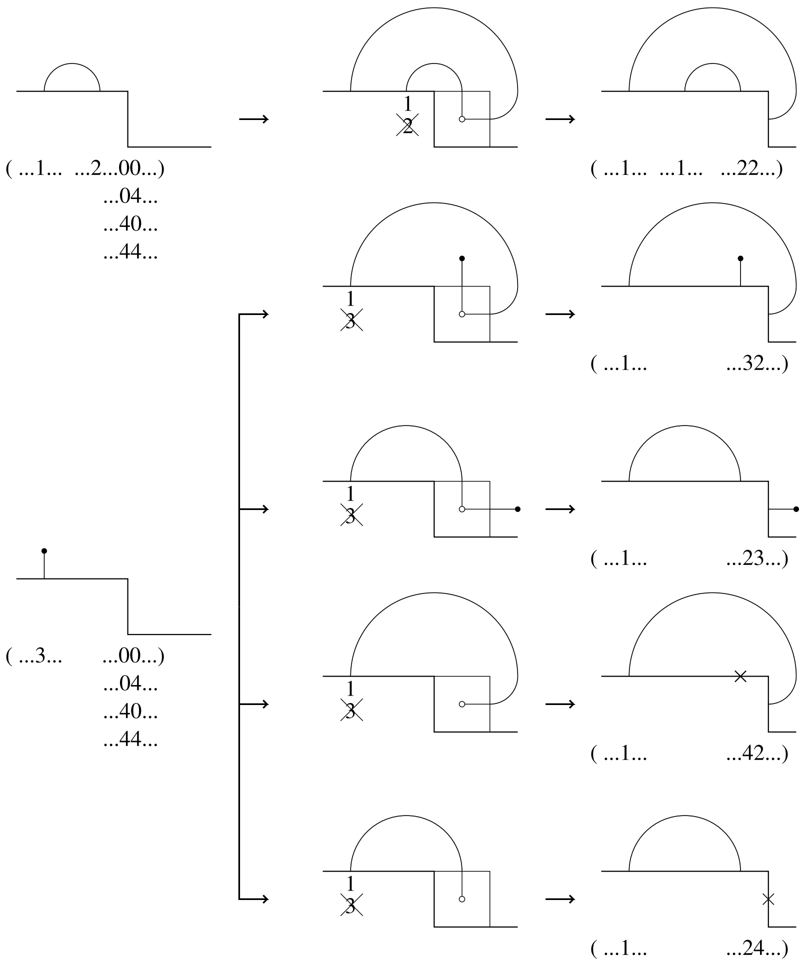

Fig. 8. No incoming lines. The monomer in the cell comes from the loops or free ends protruding out from the cut-line. The case when the corresponding loop or the free end is located at the left-hand side of the cell is depicted here. The partial chain length n increases by one. The partial contact number k remains unchanged if both of the digits associated with the empty edges are 0, increases by one if one of them is 4, or by two if both of them are 4.

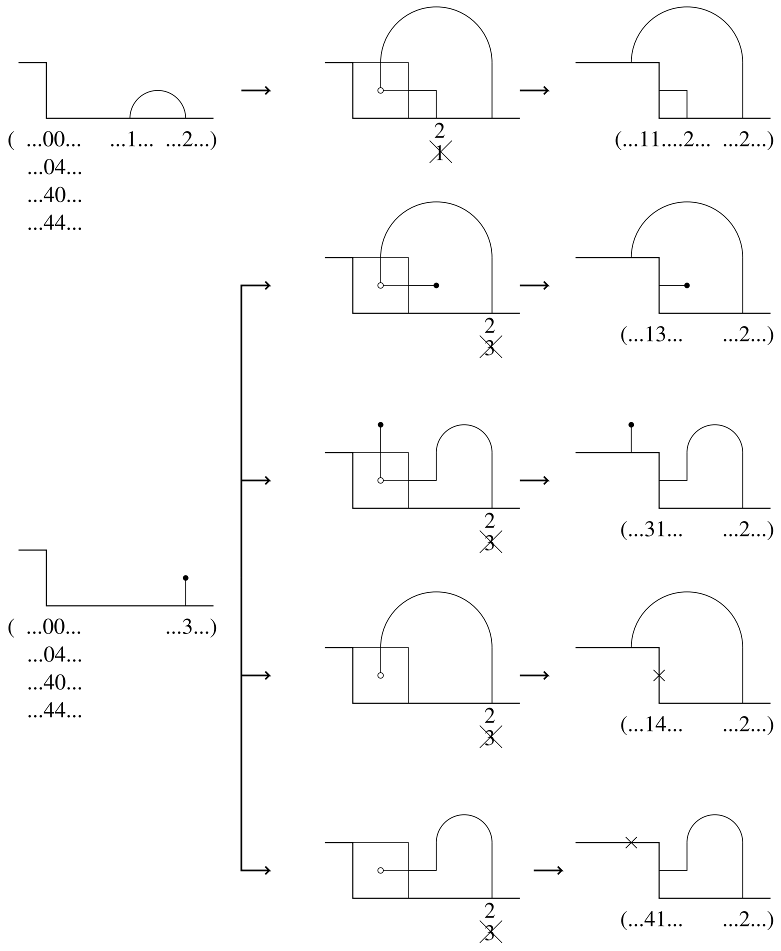

Fig. 9. No incoming lines. The monomer in the cell comes from the loops or free ends protruding out from the cut-line. The case when the corresponding loop or the free end is located at the right-hand side of the cell is depicted here. The partial chain length n increases by one. The partial contact number k remains unchanged if both of the digits associated with the empty edges are 0, increases by one if one of them is 4, or by two if both of them are 4.

from the cut-line, coming back into the cut line through the top or the right edge. The case where the corresponding loop or the free end is located at the left of the new cell is depicted in Fig. 8. In contrast to the update rules considered so far, the update rule here is non-local in that a digit far away from the edges of the current cell is also modified. Consider the case of loop joining, shown at the top of the figure. Not only does the pair (s_i, s_{i+1}) of the cut-line signature get updated to (2 2), but the digit associated with the right stem of the loop that has been joined changes from $s_k = 2$ to $s_k = 1$. Note that this ensures the balance of 1 and 2 in the new cut-line signature. Similar non-locality appears when a terminal is joined, as shown for four cases in the lower part of the figure. Again, not only does (s_i, s_{i+1}) get modified, but the digit associated with the edge where the free end is coming out, changes from $s_k = 3$ to $s_k = 1$ in the new cut-line signature, because now the protruding line forms a loop. The new pair of digits at the edges of the current cell is (2 3), (3 2), (2 4), or (4 2), depending on whether this line is entering through the right or the top edge, and whether the free end protrudes out of the current cell or terminates there, as shown in the figure. Similar update rules exist for the case when a loop or a free end at the right side of the cell passes through or terminates at the current cell (Fig. 9). Finally, the loop whose stems are at the left and the right-hand side of the current cell can be joined. In this case, only the digits associated with the current cell get modified, to $(s_i, s_{i+1}) = (2\ 1)$ (Fig. 10). The most difficult and time-consuming part of this update procedure is to find all the loops and free ends that can be joined in this manner. The method is exactly the same as that for SAW [37]. After any of these updates, the partial chain length n increases by one. The partial contact number k increases by the number of charged edges.

We note that conformations of ISAW related by discrete rotations and reflections contribute the same amount to the partition function. It is rather straightforward to remove this symmetry in the case of explicit enumeration [31,32], but this is not the case for the transfer matrix computation. Because the discrete rotational and reflectional symmetry is eight-fold [31], only a four-fold symmetry remains for the conformations spanning a non-square box if we consider only the boxes with $w < h$. We also note that in the transfer matrix method, only undirected conformations of lattice polymers are generated, whereas we want the number of conformations for directed polymers where the two directions along the polymer chain are distinguished. Therefore, for the number of conformations spanning a non-square box, we have to divide the density of states by two to obtain the symmetry-reduced density of states $\omega(K)$ for directed polymers. In the case of a square box where the rotational and reflection symmetry is eight-fold, we have to divide the result by four to obtain $\omega(K)$.

4. Pruning

In the cell-by-cell building procedure described above, many unnecessary cut-line signatures appear which cannot lead to legitimate full conformations. By removing these unnecessary states as early as possible, the computational time can be drastically reduced. In fact, the reason that the future connection topology was used for the cut-line signature in the new version of the SAW

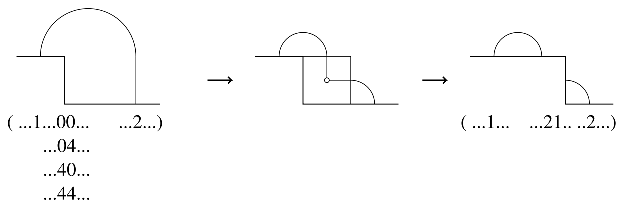

Fig. 10. Joining of the loop whose stems are located at the left- and right-hand sides of the cell. The partial chain length n increases by one. The partial contact number k remains unchanged if both of the digits associated with the empty edges are 0, increases by one if one of them is 4, or by two if both of them are 4.

transfer algorithm, rather than the past connection topology as was used in the older version, is because the specification of the future connection topology simplifies the pruning procedure considerably [37]. One can compute the minimal number of monomers required for making connections specified by the cut-line signature. Additional monomers may be needed to touch the top of the box. Also, if the value of v indicates that the partial conformation does not touch either the left or the right walls of the $w \times h$ box, then additional monomers may be required in order for the remaining part of the conformation to touch the corresponding wall. If the number of unused monomers, $N - n$, is less than the minimal required number of monomers, than the current combination of $\{s_i\}$, v, and n is pruned out and prevented from generating future conformations. The method is exactly the same as that in the case of the enumeration of SAWs [37].

In addition, if the minimal required number of monomers is less than the remaining volume of the box, or the height of the minimal-length configuration exceeds the remaining height of the box, the current state is pruned out.

5. Example

An example of the explicit step-by-step building procedure for $N = 4$ and $w \times h = 2 \times 2$ is illustrated in Fig. 11. The non-zero

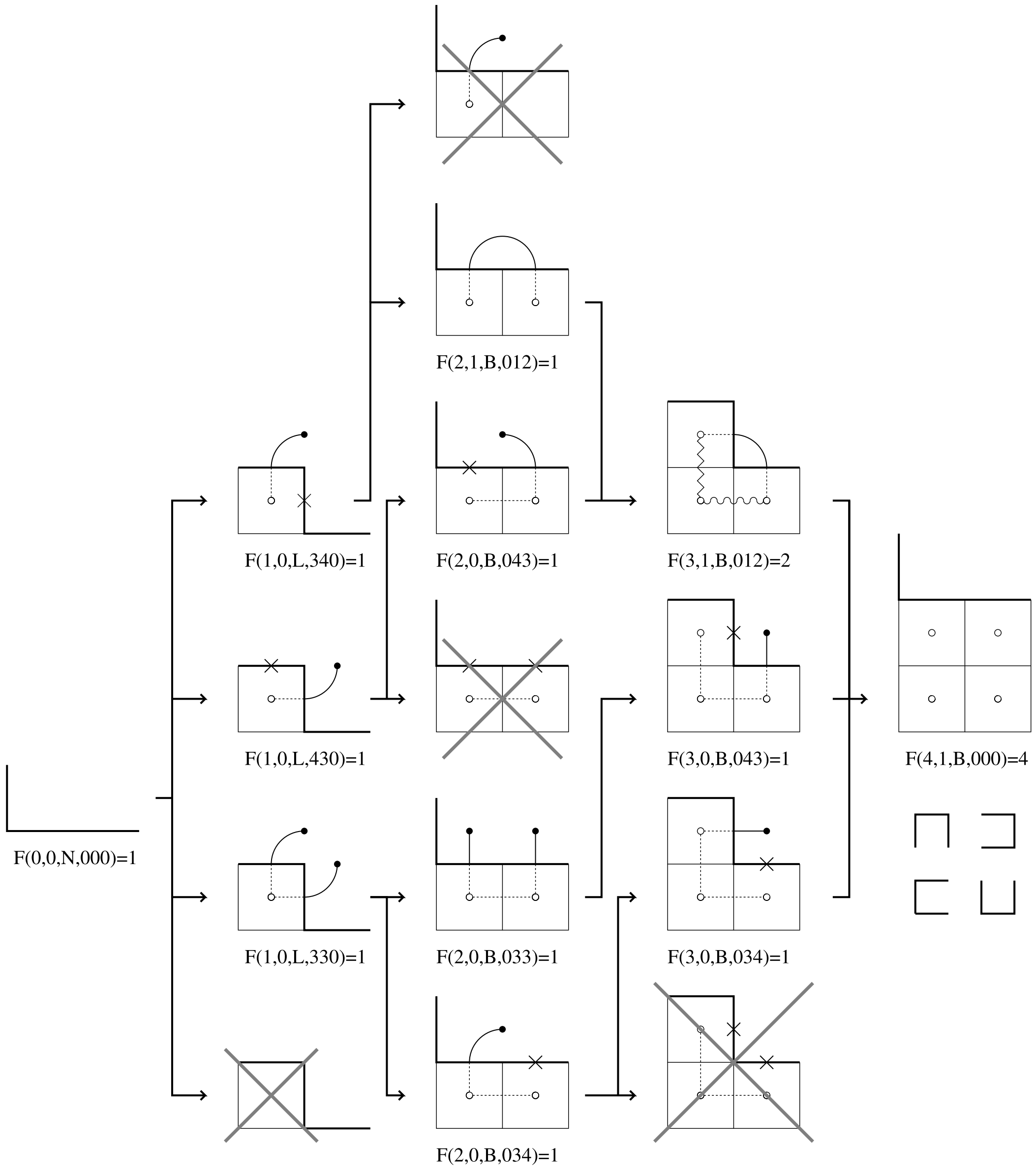

Fig. 11. An example of the step-by-step building procedure of the partial conformations, for $N = 4$ and $w \times h = 2 \times 2$. The nonzero values of $F(n, k, v, s)$ are also shown. Partial conformations that are pruned out are crossed out with grey X. Unique partial conformations below the cutline are drawn with dashed lines, and two alternative partial conformations are drawn with zigzag and wiggly lines for $(n, k, v, s) = (3, 1, B, 012)$. The four full conformations are shown below the complete 2×2 box.

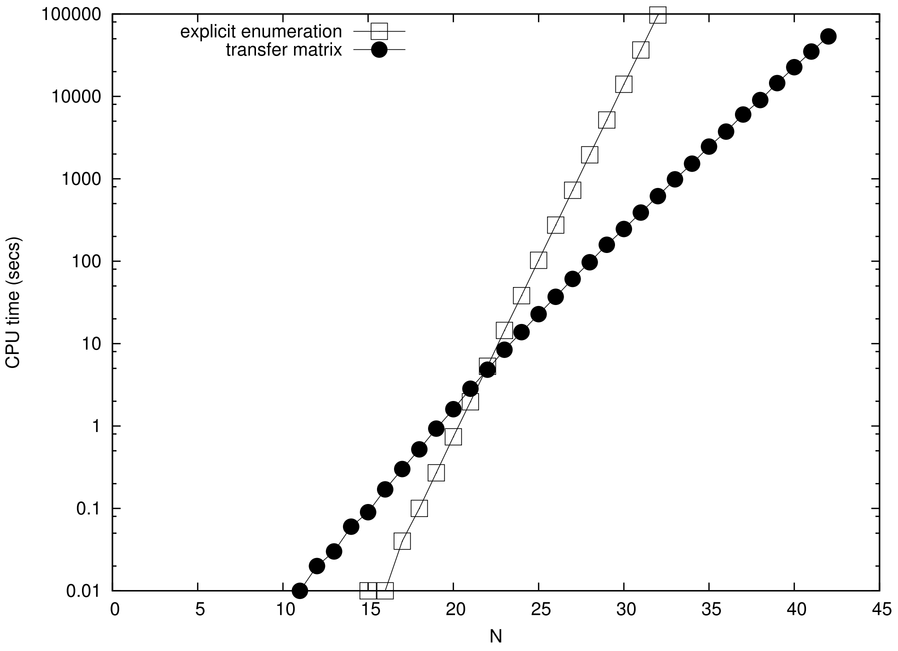

Fig. 12. Computational times of the transfer matrix and the explicit enumeration compared.

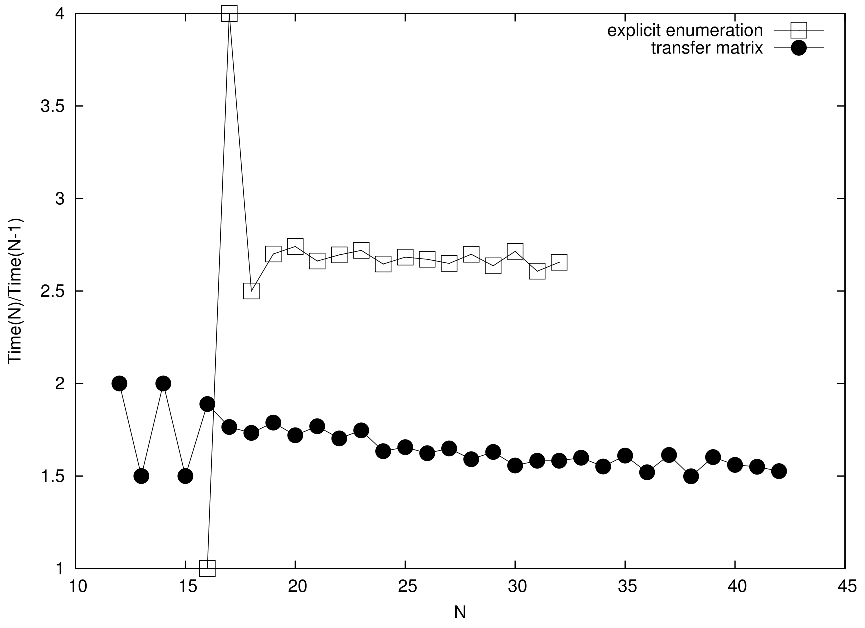

Fig. 13. Ratio of the computational times for the chain length N to that for $N - 1$, shown for both the transfer matrix computation and explicit enumeration.

values of the partial density $F(n, k, v, s)$ are also shown, where v takes the value of N, L, R, and B, depending on whether the partial conformation is touching none of the boundaries, only the left boundary, only the right boundary, or both of them, respectively. There are a total of four conformations with $K = 1$ for this box, which are related by rotations. As mentioned earlier, the density of the states has to be divided by four to remove this symmetry.

6. Computational time

In each cell of the lattice, the transfer matrix generates new cut-line signatures from the old ones. From the update rules, it is clear that the number of new cut-line signatures generated from a given old cut-line signature is of order one; therefore, the computational time for processing a cell at (i, j) will be proportional to the number of cut-line signatures prior to processing the cell, denoted as $N_s(i, j; N, w, h)$, and the total computational time $t(N, w, h)$ for computing the density of states of the polymer conformation of length N spanning the box of width w and height h will be proportional to

$$t(N, w, h) \propto \sum_{j=1}^{h} \sum_{i=1}^{w+1} N_s(i, j; N, w, h). \tag{3}$$

Because only the numbers 0, 1, 2, 3, or 4 can appear for each digit of the cut-line signatures, $N_s(i, j; N, w, h) < 5^{w+1}$. This is a strict inequality because there are actually many constraints, such as the fact that the terminals appear at most twice, the loop stems have to be balanced, etc. This bound leads to the upper bound for the

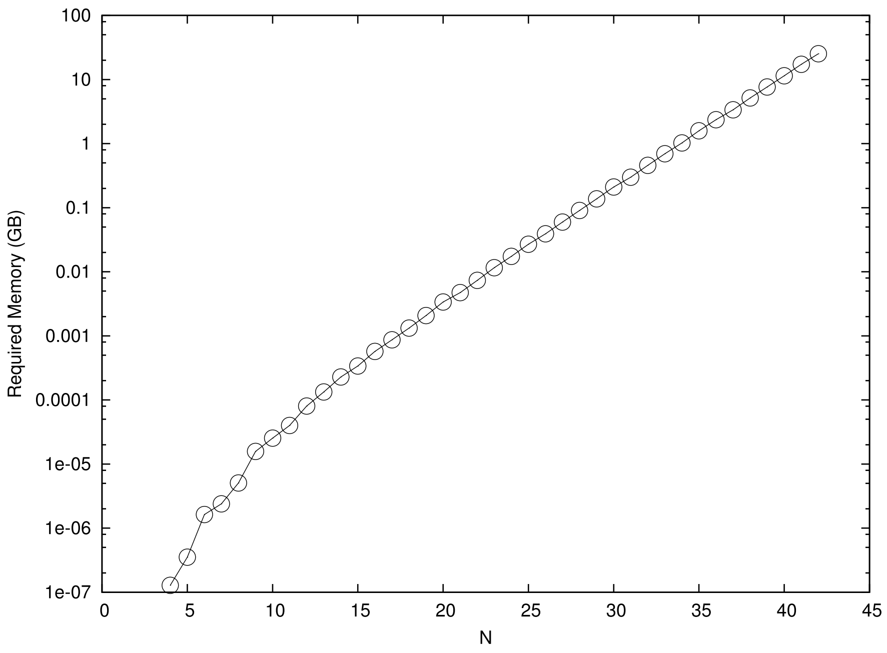

Fig. 14. Memory requirement for the partial density of states as a function of the chain length N.

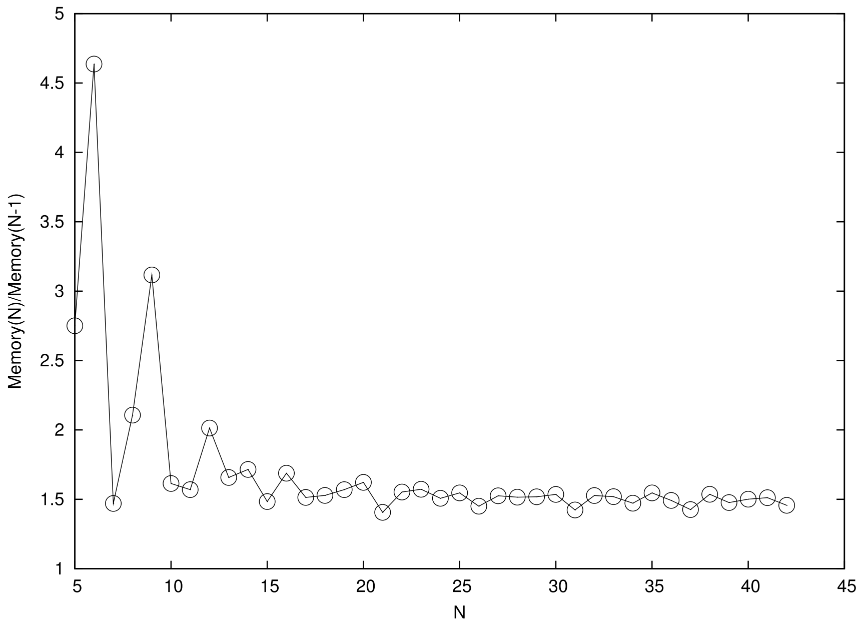

Fig. 15. Ratio of the memory size for the chain length N to that for $N-1$.

computational time:

$$t(N, w, h) < (w+1)h5^{w+1} \times \text{const}. \tag{4}$$

Because of the symmetry of the problem, we may restrict the computation to the boxes with $w \le h$ or $w \ge h$, both of which yield the same result, but the restriction to the boxes with $w \le h$ will lead to a lesser number of intermediate cut-line signatures and hence less computational time. From the conditions $w \le h$ and $w + h \le N$, we get the upper bound for w, $w \le N/2$. Therefore, the computational time $T(N)$ for the chain length N satisfies the inequality:

$$\begin{aligned} T(N) = \sum_{w,h} t(N, w, h) &< \sum_{1\le w\le N/2} N(N/2+1)5^{N/2} \times \text{const} \\ &= N^2(N+2)5^{N/2} \times \text{const} \\ &= N^2(N+2)2.24^N \times \text{const}. \end{aligned} \tag{5}$$

Asymptotically, this upper bound is better than the 2.7^N of explicit enumeration, showing that in the limit of $N \to \infty$, the transfer matrix is superior to the explicit enumeration in terms of computational time. Of course this asymptotic result may be of little use if the overall multiplicative constant in Eq. (5) is too large. Considering the worst-case scenario, it might be the case that the transfer matrix is slower for short chain lengths, and the length where the computational time of the transfer matrix method becomes comparable to that of the explicit enumeration is much longer than the range of lengths accessible to present day computers. One fortunate thing is that the upper bound in Eq. (5)

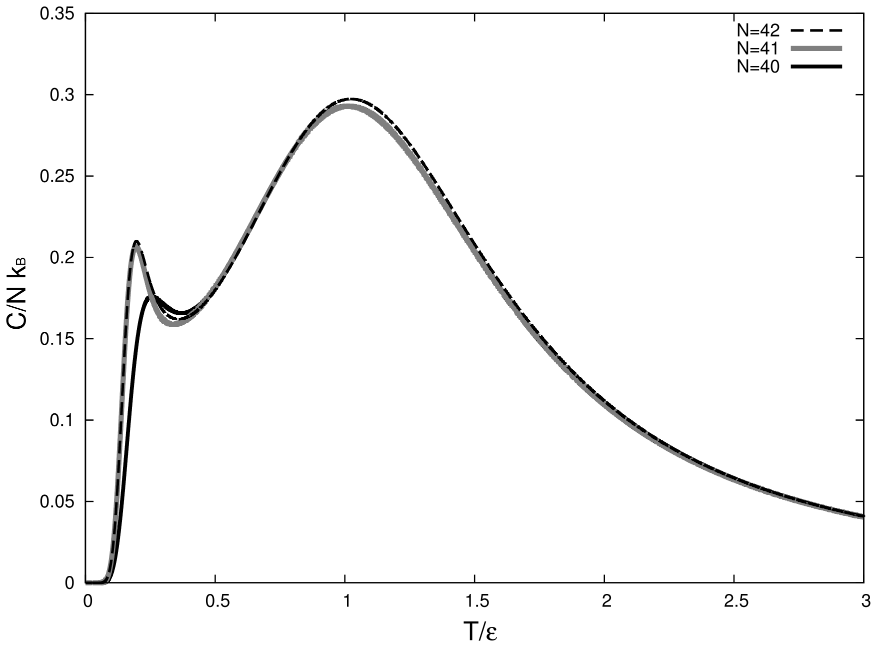

Fig. 16. Specific heat per monomer, C/Nk_B, as a function of T/ϵ, for $N = 40, 41, 42$.

is very loose, and there is a large gap between the actual $T(N)$ and this upper bound. It is crucial to reduce $T(N)$ further by pruning out unnecessary cut-line signatures in the early stages, as explained in the previous section.

The actual computational time of the transfer matrix, as well as that of the explicit enumeration, is shown in Fig. 12 as functions of the chain length, up to $N = 32$ for the explicit enumeration and up to $N = 42$ for the transfer matrix. We find that the computational time of the explicit enumeration follows the same scaling as that for the total number of conformations, and scales as $\sim 2.7^N$. On the other hand, the computational time for the transfer matrix scales as $\sim 1.6^N$. This is more clearly seen in Fig. 13 where the ratio of the computational time for the chain length N to that for $N-1$ is shown as a function of N for each method. The scaling of $\sim 1.6^N$ for the computational time of the transfer matrix is a rather conservative estimate: the ratio of the computational time for $N = 42$ to that for $N = 41$ is in fact about 1.52, suggesting that the ratio may approach 1.5 asymptotically (Fig. 13).

From these results, we see that the transfer matrix computation is faster than the explicit enumeration for $N > 21$. We note that the recent efficient implementation of explicit enumeration has increased the computational speed considerably [32]. The improvement relevant for the serial computation in a single CPU is the one-step generation of the last two monomers in the chain. This will correspond to a decrease of the effective chain length by two in terms of the computational time, resulting in the rightward horizontal shift of the graph for the explicit enumeration in Fig. 12. Taking this into account, we can safely say that the transfer matrix is faster than the explicit enumeration for $N > 25$. In fact, the computation time for chain length 42 using the transfer matrix method takes only 15 h on a *single* Intel i3-3220 CPU, whereas the explicit enumeration is expected to take about six years even if we generate the last two monomers in one step.

7. Memory requirement

In the case of the explicit enumeration, only the occupation statuses of the lattice sites are to be recorded at any moment, so the demand for the memory is virtually negligible. On the other hand, the transfer matrix requires a considerable amount of memory, because the intermediate cut-line signatures must be stored at each step. In each step, we generate the new partial densities $F_{\text{new}}(s_1, \ldots, s_{w+1}, v, n, k)$ from the old ones $F_{\text{old}}(s_1, \ldots, s_{w+1}, v, n, k)$. After the generation of F_{new}, F_{old} is no longer needed, its memory space can be recycled and be used as that for F_{new} at the next step. Therefore, we only need the memory storage for F_{new} and F_{old}.

We cannot allocate the memory space of a reasonable size for $F(s_1, \ldots, s_{w+1}, v, n, k)$ using a standard dynamic array, especially because we cannot predetermine a reasonable value of the upper bound for the number of possible signatures $(s_1, \ldots, s_{w+1})$. Therefore, we used the red–black tree [43], a data structure whose size increases as new items are inserted, which also supports fast retrievals and insertions of items, to store the combinations of $(s_1, \ldots, s_{w+1}, v)$. Because $0 \le n \le N$ and $0 \le k \le K_{\max}(N)$, the density of states may be stored as an array of size $(N+1) \times (K_{\max}(N)+1)$ for a given combination of $(s_1, \ldots, s_{w+1}, v)$. However, in order to save further memory space, we stored the partial density of states only for the values of n with a non-zero number of conformations, using the linked list. The partial density of states $F(s_1, \ldots, s_{w+1}, v, n, k)$ for a given combination of $(s_1, \ldots, s_{w+1}, v, n)$ was then stored as an array of size $K_{\max}(N)+1$. Because $F(s_1, \ldots, s_{w+1}, v, n, k)$ is stored as an eight-byte unsigned long integer, the memory requirement N_{mem} in bytes is

$$N_{\text{mem}} = 16 N_c^{\max}(K_{\max}(N)+1), \tag{6}$$

neglecting the space for other variables. Here, $N_c^{\max}$ denotes the maximum number of combinations $(s_1, \ldots, s_{w+1}, v, n)$ encountered during the progression of the algorithm. The extra factor of two comes from the fact that memory space for both F_{new} and F_{old} is required [1]. The values of $(K_{\max}+1)$, $N_c^{\max}$, $(K_{\max}+1)\cdot N_c^{\max}$, and N_{mem} are given in Table 1 as functions of N, and the graph for N_{mem} is shown in Fig. 14. The memory asymptotically scales as $\sim 1.5^N$, as can be seen from the graph for the ratio of memory space for chain length N to that for $N-1$ (Fig. 15). A chain length of up to

[1] Of course the numbers of the old and the new signatures are different at any moment in general, so the memory space needed will be slightly less than that given in Eq. (6) but we neglect this small difference.

Table 1
Memory space required, N_{mem}, as functions of the chain length N. The factors contributing to N_{mem} are also shown.

N	$K_{\max}+1$	$N_c^{\max}$[a]	$(K_{\max}+1)\cdot N_c^{\max}$	N_{mem}[b]
3	1	0	0	0
4	2	4	8	128
5	2	11	22	352
6	3	34	102	1 632
7	3	50	150	2 400
8	4	79	316	5 056
9	5	197	985	15 760
10	5	318	1 590	25 440
11	6	416	2 496	39 936
12	7	718	5 026	80 416
13	7	1 190	8 330	133 280
14	8	1 786	14 288	228 608
15	9	2 355	21 195	339 120
16	10	3 577	35 770	572 320
17	10	5 413	54 130	866 080
18	11	7 520	82 720	1 323 520
19	12	10 819	129 828	2 077 248
20	13	16 196	210 548	3 368 768
21	13	22 768	295 984	4 735 744
22	14	32 820	459 480	7 351 680
23	15	48 165	722 475	11 559 600
24	16	68 046	1 088 736	17 419 776
25	17	99 033	1 683 561	26 936 976
26	17	143 609	2 441 353	39 061 648
27	18	206 856	3 723 408	59 574 528
28	19	296 976	5 642 544	90 280 704
29	20	428 236	8 564 720	137 035 520
30	21	626 008	13 146 168	210 338 688
31	21	890 622	18 703 062	299 248 992
32	22	1 298 532	28 567 704	457 083 264
33	23	1 886 902	43 398 746	694 379 936
34	24	2 662 054	63 889 296	1 022 228 736
35	25	3 951 437	98 785 925	1 580 574 800
36	26	5 669 758	147 413 708	2 358 619 328
37	26	8 082 368	210 141 568	3 362 265 088
38	27	11 957 233	322 845 291	5 165 524 656
39	28	17 019 325	476 541 100	7 624 657 600
40	29	24 664 128	715 259 712	11 444 155 392
41	30	36 042 443	1 081 273 290	17 300 372 640
42	31	50 797 197	1 574 713 107	25 195 409 712

[a] The maximal number of combinations $(s_1, \ldots, s_{w+1}, n)$, encountered during the progression of the algorithm.
[b] The memory requirement in bytes.

$N = 46$ seems feasible with 128 GB of memory. Memory space can be further saved by storing only the non-zero partial density of states for a given combination of $(s_1, \ldots, s_{w+1}, v, n)$, instead of using an array of fixed size $K_{\max}(N)+1$. We have not implemented this additional flexibility at this stage, because it will make the code unnecessarily complicated.

8. New results

With the transfer matrix, all the known results for up to $N = 40$ could be reproduced [31,32,44], and the new results for $N = 41$ and $N = 42$ could be obtained. The symmetry reduced densities of states $\omega(K)$ are shown for $N = 41$ and $N = 42$ in Table 2. The correctness of the result can be also cross checked against the total number of SAWs, $\sum_K \Omega(K)$, previously obtained using a transfer matrix algorithm for up to $N = 80$ [37,38,45].

It is straightforward to compute the exact partition function from the density of states using the formula Eq. (1), from which various physical quantities can be obtained. One example of such a quantity is the specific heat per monomer:

$$\frac{C}{k_B N} = \frac{1}{N k_B}\frac{\partial \langle E\rangle}{\partial T} = \frac{1}{N k_B^2 T^2}\left(\frac{\partial^2 \ln Z}{\partial \beta^2}\right)$$

Table 2
Densities of states for $N = 41$ and $N = 42$. The conformations related by rotation or reflection are counted only once.

K	$N = 41$	$N = 42$
0	204 215 004 596 272	476 389 994 800 229
1	885 251 445 177 512	2 115 847 261 636 492
2	2 079 694 461 161 427	5 084 598 808 157 791
3	3 464 902 826 469 576	8 657 251 498 020 670
4	4 595 250 741 229 180	11 720 220 074 174 806
5	5 156 290 616 308 466	13 411 555 823 963 447
6	5 083 784 125 308 556	13 473 468 289 589 989
7	4 512 952 758 682 396	12 179 635 324 640 045
8	3 670 155 319 845 554	10 081 627 576 356 447
9	2 768 628 794 352 198	7 738 288 654 944 187
10	1 955 769 586 807 773	5 560 932 203 029 585
11	1 302 911 413 863 672	3 768 601 894 558 080
12	823 108 147 575 701	2 422 174 369 581 956
13	495 137 600 489 206	1 482 830 916 877 836
14	284 503 636 510 917	867 529 474 495 897
15	156 477 901 312 440	486 206 027 920 648
16	82 492 824 518 467	261 467 916 938 889
17	41 702 482 928 294	135 033 618 385 132
18	20 209 523 024 356	66 985 846 124 393
19	9 372 380 538 742	31 891 765 478 273
20	4 149 991 633 601	14 547 520 272 987
21	1 746 712 880 458	6 339 793 465 387
22	693 139 648 771	2 627 364 285 081
23	257 015 560 326	1 027 082 613 128
24	87 861 707 542	375 429 560 638
25	26 600 160 006	125 822 825 988
26	6 879 377 897	37 442 098 467
27	1 268 269 356	9 537 009 150
28	95 375 740	1 636 035 133
29	744 882	106 244 025
30		810 017
Total	37 599 781 156 059 284	100 047 629 074 894 793

$$= \frac{1}{N k_B^2 T^2}\left[\frac{\sum_q q^2 \Omega(q) e^{\beta q \epsilon}}{\sum_p \Omega(p) e^{\beta p \epsilon}} - \left(\frac{\sum_q q \Omega(q) e^{\beta q \epsilon}}{\sum_p \Omega(p) e^{\beta p \epsilon}}\right)^2\right]. \tag{7}$$

The specific heats for $40 \le N \le 42$ are shown in Fig. 16 as functions of the temperature T/ϵ. The two peaks at $T/\epsilon \simeq 1.0$ and $T/\epsilon \simeq 0.2$ correspond to the collapse and freezing transitions, respectively [27–29]. As can be seen in the figure, in contrast to the peak for the collapse transition that changes smoothly with N, the peak for the freezing transition becomes especially prominent as N approaches 42. This is because $42 = 6 \times 7$ is the magic number where the ground states have a special form and their numbers are smaller than for neighbouring values of N [27,28,46]. There is a possibility that this transition is only a finite-size effect and does not exist in the infinite-size limit. The study of polymers with longer chain sizes may shed more light on this issue.

9. Discussion

In this work, I developed a transfer matrix method for computing the exact partition function of the ISAW model on the square lattice, by extending the previous algorithm for the enumeration of the total number of SAW conformations [37,38]. We found that the computational time scales as 1.6^N, in contrast to 2.7^N in the case of explicit enumeration, and all the densities of states for chain lengths of up to $N = 42$ could be obtained within two days with a *single* CPU.

However, the transfer matrix method developed in this work is not meant to replace the explicit enumeration. Rather, the transfer matrix method is a tool which is complementary to the explicit enumeration. In its current form, the method can be used only for the square-lattice homopolymer, and cannot be applied to other lattice models such as three-dimensional polymer or HP protein models. Even in the context of the square-lattice homopolymer, only the quantities solely determined by the spatial distribution of the monomers, regardless of their positions *along the chain*, can be computed by the present method. The contact numbers considered in the current work, and other geometrical quantities such as radius of gyration [37,38], are such examples. On the other hand, if we want to compute the average length of monomers connecting two monomers that are in spatial contact, the transfer matrix method cannot be used, because such non-local information about the chain is not maintained in the building process of conformations. Another limitation is the requirement of memory resources, which is virtually zero in the case of explicit enumeration. Therefore, in these situations where the transfer matrix method does not work, explicit enumeration remains a valuable tool.

The tremendous amounts of computational time required for the explicit enumeration of polymer chains have been overcome by parallelization [32,36]. We can expect the same for the transfer matrix algorithms. For the algorithm developed in the current work, the simplest method for parallelization would be to distribute the boxes to the nodes, because the enumerations of conformations spanning distinct boxes are independent tasks. The same idea has been used in the context of explicit enumeration [31]. The efficiency of this simple method is far from ideal, because the number of conformations varies greatly from box to box, and some of the nodes will keep working while the others have already finished the task. It is crucial to distribute the load evenly among the computational nodes to obtain maximal efficiency. In fact, an efficient parallel implementation of the transfer matrix method for enumerating self-avoiding polygon (SAP) has been developed where cut-line signatures rather than boxes are distributed among the nodes [47]. This idea is also similar to the recent efficient parallelization of the explicit enumeration where the partial conformations rather than the boxes are distributed among the nodes [32]. Because the method developed in the current work is an extension of the method for SAW enumeration [37,38], which is in turn an extension of the method for SAP [40,41], and thus shares a common backbone structure, it is in principle straightforward to implement such parallelization. It is expected that the parallelization based on the distribution of cut-line signatures will also reduce the memory burden of each node, allowing us to compute the density of states for longer chains with ease.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF-2017R1D1A1B03031344).

References

[1] H.S. Chan, K.A. Dill, Macromolecules 22 (1989) 4559.
[2] H.S. Chan, K.A. Dill, Annu. Rev. Biophys. Biophys. Chem. 20 (1991) 447.
[3] P.J. Flory, Principles of Polymer Chemistry, Cornell University Press, Ithaca, 1967.
[4] P.-G. de Gennes, Scaling Concepts in Polymer Physics, Cornell University Press, Ithaca, 1979.
[5] M.J. Stephen, Phys. Lett. A 53 (1975) 363.
[6] A. Baumgärtner, J. Physique 43 (1982) 1407.
[7] T. Ishinabe, J. Chem. Phys. 77 (1982) 3171;
T. Ishinabe, J. Chem. Phys. 80 (1984) 1318;
T. Ishinabe, J. Phys. A 18 (1985) 3181.
[8] A.L. Kholodenko, K.F. Freed, J. Phys. A 17 (1984) L191;
A.L. Kholodenko, K.F. Freed, J. Chem. Phys. 80 (1984) 900.
[9] T.M. Birshtein, S.V. Buldyrev, A.M. Elyashevitch, Polymer 26 (1985) 1814.
[10] B. Derrida, H. Saleur, J. Phys. A 18 (1985) L1075.
[11] V. Privman, J. Phys. A 19 (1986) 3287.
[12] H. Saleur, J. Stat. Phys. 45 (1986) 419.
[13] B. Duplantier, H. Saleur, Phys. Rev. Lett. 59 (1987) 539.
[14] F. Seno, A.L. Stella, J. Phys. France 49 (1988) 739.
[15] P.H. Poole, A. Coniglio, N. Jan, H.E. Stanley, Phys. Rev. B 39 (1989) 495.
[16] H. Meirovitch, H.A. Lim, Phys. Rev. Lett. 62 (1989) 2640.
H. Meirovitch, H.A. Lim, J. Chem. Phys. 91 (1989) 2544.
[17] I. Chang, H. Meirovitch, Phys. Rev. E 48 (1993) 3656.
[18] P. Grassberger, R. Hegger, J. Phys. I France 5 (1995) 597.
[19] G.T. Barkema, U. Bastolla, P. Grassberger, J. Stat. Phys. 90 (1998) 1311.
[20] S.L. Narasimhan, P.S.R. Krishna, K.P.N. Murthy, M. Ramanadham, Phys. Rev. E 65 (2001) 010801(R).
[21] C.-N. Chen, C.-Y. Lin, Physica A 350 (2005) 45.
[22] J. Zhou, Z.-C. Ou-Yang, H. Zhou, J. Chem. Phys. 128 (2008) 124905.
[23] A.G. Cunha-Netto, R. Dickman, A.A. Caparica, Comput. Phys. Comm. 180 (2009) 583.
[24] M. Gaudreault, J. Viñals, Phys. Rev. E 80 (2009) 021916.
[25] S. Caracciolo, M. Gherardi, M. Papinutto, A. Pelissetto, J. Phys. A 44 (2011) 115004.
[26] M. Ponmurugan, S.V.M. Satyanarayana, J. Stat. Mech. (2012) P06010.
[27] T. Vogel, M. Bachmann, W. Janke, Phys. Rev. E 76 (2007) 061803.
[28] T. Wüst, D.P. Landau, Phys. Rev. Lett. 102 (2009) 178101.
[29] C.-N. Chen, Y.-H. Hsieh, C.-K. Hu, Europhys. Lett. 104 (2013) 20005.
[30] J.H. Lee, S.-Y. Kim, J. Lee, J. Chem. Phys. 133 (2010) 114106;
J.H. Lee, S.-Y. Kim, J. Lee, J. Chem. Phys. 135 (2011) 204102;
J.H. Lee, S.-Y. Kim, J. Lee, Phys. Rev. E. 86 (2012) 011802;
J.H. Lee, S.-Y. Kim, J. Lee, Phys. Rev. E. 87 (2013) 052601;
J.H. Lee, S.-Y. Kim, J. Lee, AIP Adv. 6 (2016) 055013.
[31] J.H. Lee, S.-Y. Kim, J. Lee, Comput. Phys. Comm. 182 (2011) 1027.
[32] Y.-H. Hsieh, C.-N. Chen, C.-K. Hu, Comput. Phys. Comm. 209 (2016) 27.
[33] M. Suzuki, Phys. Rev. B 3 (1985) 2957.
[34] S.-Y. Kim, R.J. Creswick, Phys. Rev. E 58 (1998) 7006.
[35] P. Bruscolini, A. Pelizzola, Phys. Rev. Lett. 88 (2002) 258101.
[36] J. Lee, Phys. Rev. Lett. 110 (2013) 248101;
J. Lee, Phys. Rev. E 88 (2013) 022710.
[37] I. Jensen, 2013. arXiv:1309.6709 [math-ph].
[38] I. Jensen, J. Phys. A: Math. Gen. 37 (2004) 5503.
[39] A.R. Conway, I.G. Enting, A.J. Guttmann, J. Phys. A: Math. Gen. 26 (1993) 1519.
[40] N. Clisby, I. Jensen, J. Phys. A 45 (2012) 115202.
[41] I. Jensen, A.J. Guttmann, J. Phys. A: Math. Gen. 32 (1999) 4867.
[42] I.G. Enting, J. Phys. A: Math. Gen. 13 (1980) 3713.
[43] R. Bayer, Acta Inform. 1 (1972) 290.
[44] J.H. Lee, S.-Y. Kim, J. Lee, J. Phys.: Conf. Ser. 454 (2013) 012083.
[45] http://www.ms.unimelb.edu.au/~ij@unimelb/saw/series/sqsaw.ser.
[46] J.H. Lee, J. Lee, S.-Y. Kim, J. Korean Phys. Soc. 66 (2015) 1797; J. Korean Phys. Soc. 69 (2016) 1518.
[47] I. Jensen, J. Phys. A: Math. Gen. 36 (2003) 5731.