


Guided Research Project

Implementation of a Classical Simulation of Quantum Cellular Automata on 1-D Lattices and Using it to Improve the Tensor Network Algorithm TDVP

Benjamin Decker  

 benjamin.decker@tum.de

April 26, 2023

Abstract — The 2-site TDVP algorithm modifies MPS bond-dimensions on the fly, speeding up when entanglement is low. However, this comes at the cost of slowing down simulations when entanglement is high. The most time consuming part of the TDVP is solving the Schrödinger Equation for its effective Hamiltonians. In the 2-site variant, the effect of this is made worse by the increased dimension of combined 2-site tensors. However, limiting the time evolution to 1-site tensors complicates deciding when to grow the bond dimensions.

This article presents a new version of the TDVP algorithm, inspired by the existing A1TDVP algorithm which enables dynamic bond dimensions while avoiding to solve 2-site SEs. Like the A1TDVP, the new algorithm is also capable of changing bond dimensions on the fly, without solving SEs on combined tensors. By relying on Singular Value Decompositions and approximate time evolutions, accurate estimations for good bond dimensions can be calculated

However, in the current NISQ era, without the help of powerful quantum computers [6], quantum systems like QCAs have to be simulated on classical hardware. Unfortunately, the vast memory and computing power required to classically represent quantum systems sets constraints on what can be simulated and what is unfeasible, see Section 2.5. That’s why efficient algorithms like the time-dependent variational principle (TDVP), see Section 4, are needed to alleviate the issue. This article presents the new 1TDVP (n1TDVP), a new version of the TDVP. The goal of the n1TDVP is to improve on the time efficiency of existing TDVP-versions, without sacrificing accuracy.

Alongside this paper, I implemented a classical simulation of Quantum Cellular Automata which was used for the simulations shown in Section 5.1. The new n1TDVP emerged as an idea during the implementation of the simulation, so all simulations in this paper are limited to QCA. The code of the project is available on GitHub [7].

1 Introduction

Cellular automata (CAs) have historically been a surprisingly plentiful source of research, considering the simplicity of their definition. The usefulness of CAs ranges over many different fields, with applications in cryptography [1], molecular dynamics [2], and chemistry [3], to just name a few. As a consequence, every advancement in the study of CAs potentially impacts advancements in many other connected fields. The incentive is therefore high to improve on the concept.

One such improvement are quantum cellular automata (QCAs), which translate operations of a classical CA into the frame of quantum computing. By leveraging the computational power and information density of quantum systems, QCAs will potentially reveal new insights into the world of CAs, or even pose a completely new model with previously unknown capabilities [4]. Already QCAs are showing potential for simulating quantum computers or modelling a system of quantum particles [5].

2 Quantum Cellular Automata

A CA consists of a regular grid in an arbitrary dimension, a finite amount of states each grid point can be in, and a set of rules that determine if and how the state of a grid point changes depending on its current state and the states of its neighbor cells. Given an initial state of the grid, the automaton can then be simulated step by step, often displaying unexpectedly complex behavior.

2.1 Classical Cellular Automata

One of the best known examples of a classical CA is Conway’s Game of Life, named after its discoverer John Horton Conway in 1970 [8]. It consists of a two-dimensional grid with two possible grid states “alive” and “dead”, and a set of rules inspired by cells dying or coming alive due to under- or overpopulation in their immediate neighborhood.

There are a lot of known initial states in the Game of Life with predictable behavior, like the “pulsar” with a

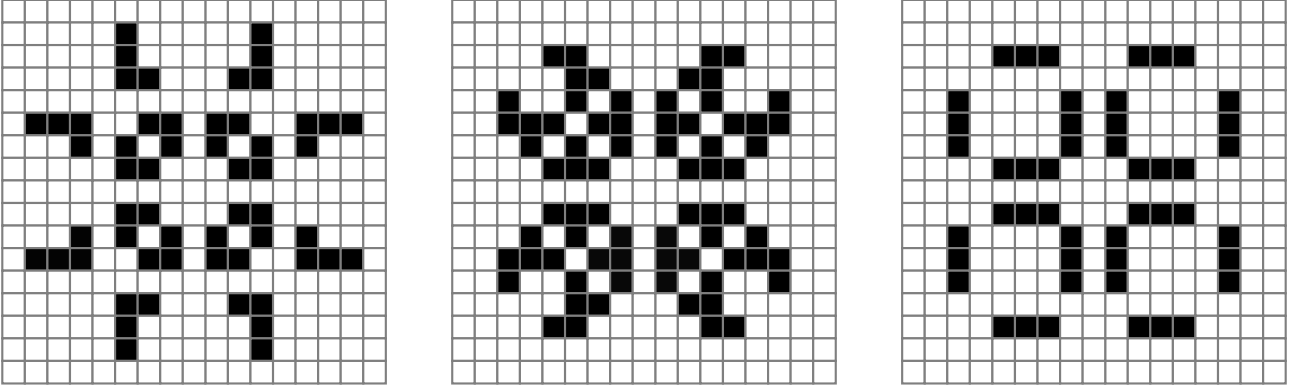


Figure 1 All 3 states of a pulsar with period 3 in Conway's Game of Life, with time advancing from left to right

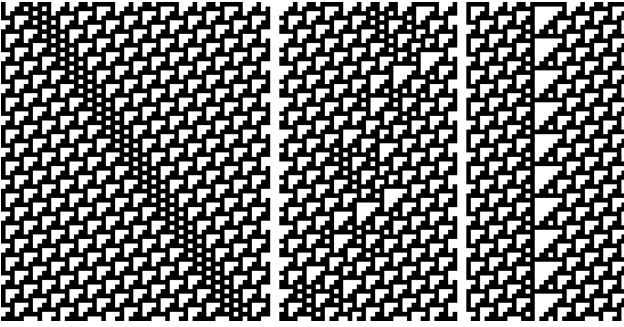


Figure 2 A glider in the rule-110 elementary cellular automaton with space along the x-axis and time along the y-axis Image source: [10]

period of three time steps shown in Figure 1. However, for random initial states the time evolution generally does not converge to static or periodic behavior and is therefore not easily predictable [9].

Conway's Game of Life and other two-dimensional CAs are great examples of how complex behavior can arise from simple rules. An even simpler, but not less interesting, class of CAs is found by limiting the scope to one dimension, where the same surprising complexity as in higher dimensions can be found. Figure 2 shows a glider in a one-dimensional automaton.

As it turns out, nearest-neighbor interactions on a one-dimensional chain like this have similarities to quantum Ising-models [11]. Inspired by these similarities, a quantum-analogue of a cellular automaton can be created.

2.2 Quantum State

Moving from classical CAs to QCAs, the first goal is to represent the automaton state as a quantum state vector. To find an quantum-analogue of a classical 1-dimensional chain of cells, an obvious choice is to work with states on a 1-dimensional chain of N -

quantum systems. By choosing $|0\rangle$ and $|1\rangle$ as basis states with

$$|0\rangle := \text{dead}, \quad |1\rangle := \text{alive}, \quad (1)$$

the full system state $|\psi\rangle$ exists in a Hilbert space with dimension 2^N , where N is the number of cells [4]. For a given initial state, the state can be expressed via the Kronecker product [12]:

$$\begin{aligned} 000101000 &\longrightarrow |0\rangle^{\otimes 3} \otimes |1\rangle \otimes |0\rangle \otimes |1\rangle \otimes |0\rangle^{\otimes 3} \\ &\implies |\psi\rangle = |000101000\rangle. \end{aligned} \quad (2)$$

To compute the probability of a cell at index j being alive or dead, the corresponding observables

$$\hat{n}_j = |0\rangle_j \langle 0| \quad \hat{n}_j = |1\rangle_j \langle 1|, \quad (3)$$

are used to measure the system state [13]:

$$\begin{aligned} P(\text{dead})_j &= \langle \psi | \hat{n}_j | \psi \rangle \\ P(\text{alive})_j &= \langle \psi | \hat{n}_j | \psi \rangle. \end{aligned} \quad (4)$$

2.3 Hamiltonian

Additionally, a unitary time evolution following the analogue of classical automaton rules must be created. An intuitive way of achieving this is by creating a Hamiltonian first. The Hamiltonian is constructed based on the set of rules the QCA should follow. To change a given cell j of the chain according to some chosen rule, the Hamiltonian governing this change can be constructed with an operation gate \hat{S}_j and a projection operator \hat{P}_j [13]:

$$\hat{H}_j = \hat{S}_j \hat{P}_j. \quad (5)$$

Operator \hat{P}_j is chosen "to be different from zero over the set of states where" [13] the rule should act on cell j , and \hat{S}_j describes the change that happens in this case.

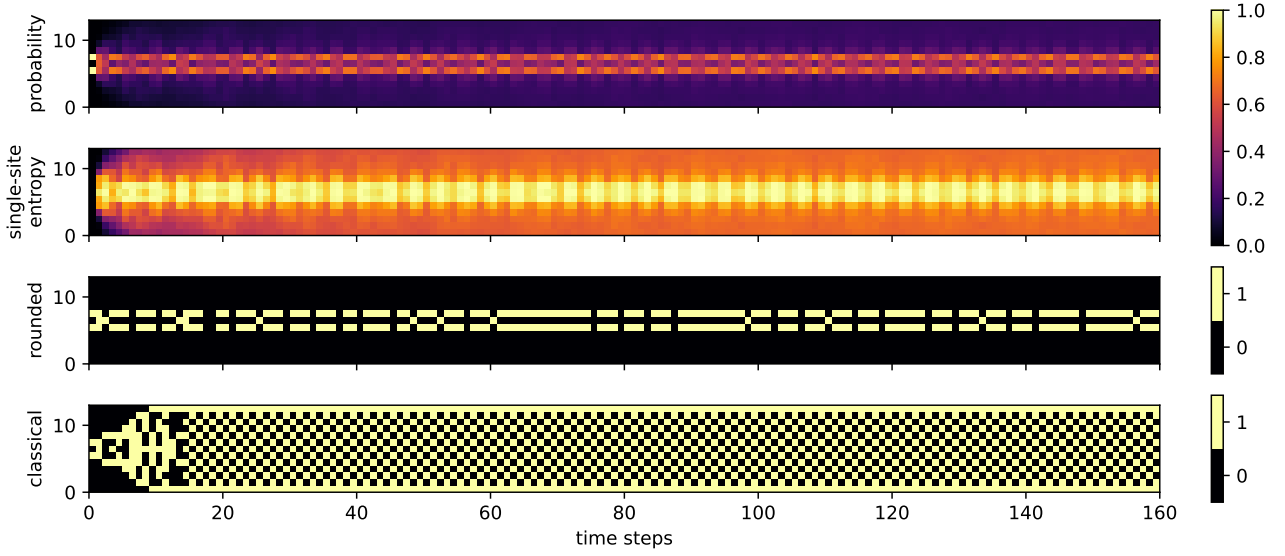


Figure 3 Time evolution of the initial state $|\psi\rangle_0 = |0\rangle^{\otimes 5} \otimes |101\rangle \otimes |0\rangle^{\otimes 5}$ according to Equation (12), showing the probability that a cell is alive (first), the single-site entropy of a cell vs. the rest of the state (second), a rounded version of the probability plot (third), and a non-quantum evolution according to classical rules (fourth)

The complete Hamiltonian can then be constructed by taking a sum over all single-site Hamiltonians:

$$\hat{H} = \sum_j \hat{S}_j \hat{P}_j. \quad (6)$$

One possible set of rules in a one-dimensional classical automaton is called F_{12} [13]:

$F_{12} :=$ A cell is flipped iff the number of alive cells among its nearest and next-nearest neighbors is exactly 2 or 3.

The projection operator \hat{P}_j for F_{12} can be constructed from

$$\begin{aligned} \hat{N}_j^{(2)} = & \hat{n}_{j-2} \hat{n}_{j-1} \hat{n}_{j+1} \hat{n}_{j+2} + \hat{n}_{j-2} \hat{n}_{j-1} \hat{n}_{j+1} \hat{n}_{j+2} \\ & + \hat{n}_{j-2} \hat{n}_{j-1} \hat{n}_{j+1} \hat{n}_{j+2} + \hat{n}_{j-2} \hat{n}_{j-1} \hat{n}_{j+1} \hat{n}_{j+2} \\ & + \hat{n}_{j-2} \hat{n}_{j-1} \hat{n}_{j+1} \hat{n}_{j+2} + \hat{n}_{j-2} \hat{n}_{j-1} \hat{n}_{j+1} \hat{n}_{j+2} \end{aligned} \quad (7)$$

$$\begin{aligned} \hat{N}_j^{(3)} = & \hat{n}_{j-2} \hat{n}_{j-1} \hat{n}_{j+1} \hat{n}_{j+2} + \hat{n}_{j-2} \hat{n}_{j-1} \hat{n}_{j+1} \hat{n}_{j+2} \\ & + \hat{n}_{j-2} \hat{n}_{j-1} \hat{n}_{j+1} \hat{n}_{j+2} + \hat{n}_{j-2} \hat{n}_{j-1} \hat{n}_{j+1} \hat{n}_{j+2}, \end{aligned} \quad (8)$$

which are non-zero over the set of states where 2 or 3 cells are alive, respectively [13]. The flip itself is done by the Pauli-X gate on cell j :

$$\sigma_j^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}_j. \quad (9)$$

With everything in place, the Hamiltonian governing the F_{12} rule can be assembled, following Equation (6) [13]:

$$\hat{H}_{F_{12}} = \sum_{j=3}^{L-2} \sigma_j^x \left(\hat{N}_j^{(2)} + \hat{N}_j^{(3)} \right). \quad (10)$$

This Hamiltonian describes the desired property for all cells, except the outermost and second-outermost ones, which are excluded because they don't have enough neighbors. These cells are controlled by separate terms with reduced versions of the projection operators from Equations (7) and (8) that only take the existing neighbors into account.

2.4 Time Evolution Operator

With the finished Hamiltonian, a unitary time evolution operator can be derived by solving the Schrödinger equation (SE) [12]

$$i\partial_t |\psi\rangle = \hat{H} |\psi\rangle \longrightarrow \hat{U}(t) = e^{-i\hat{H}t}, \quad (11)$$

and using $\hat{U}(t)$ to evolve an initial state $|\psi\rangle_t$ over time t into a state $|\psi\rangle_t$:

$$|\psi\rangle_t = \hat{U}(t) |\psi\rangle_0. \quad (12)$$

The time t needed for a state to flip under the action of operator σ_{xi} from Equation (10) is $\frac{\pi}{2}$, so since the automaton was inspired by its classical counterpart where one flip also takes one time step to complete, t is set to $\frac{\pi}{2}$ in Equations (11) and (12). By evolving an initial state $|\psi\rangle_0$ with $\hat{U}(t)$ multiple times and measuring the updated state after each step, as described in Equation (4), the evolution of the state can be observed and compared to its classical counterpart. The result is plotted in Figure 3.

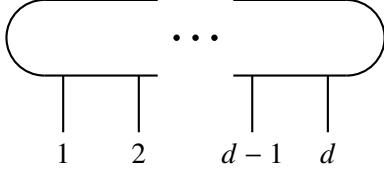


Figure 4 A tensor $\in \mathbb{C}^{n_1 \times n_2 \times \dots \times n_{d-1} \times n_d}$ of order d

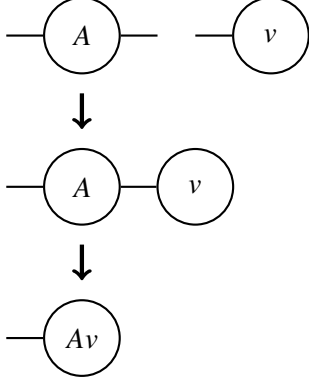


Figure 5 Creation and contraction of a tensor network representing a matrix-vector multiplication Av with $A \in \mathbb{C}^{m \times n}$ and $v \in \mathbb{C}^n$

2.5 Curse of Dimensionality

Calculating an exact solution to the SE is fast and simple for a small number of cells. However, due to the dimension of the Hilbert space growing exponentially with the number of cells, this method quickly becomes infeasible at more than around 13 cells, depending on hardware capabilities.

The general reason for the exponentially growing size of Hilbert spaces describing quantum systems is the possible entanglement between system states [12]. However, for many simulations only a small set of states show a significant measure of correlation. This results in redundancies that can be leveraged to efficiently represent a close approximation of the original system. As a consequence, many simulations can be approximated with negligible errors in exponentially smaller subspaces, compared to an exact time evolution approach, thus making simulations with an arbitrary number of cells viable again. One such method are tensor networks and corresponding tensor network algorithms.

3 Tensor Networks

Tensors are “generalizations of vectors and matrices to multi-dimensional arrays” [14] of the form

$$T = t_{j_1, j_2, \dots, j_{d-1}, j_d} \in \mathbb{C}^{n_1 \times n_2 \times \dots \times n_{d-1} \times n_d}, \quad (13)$$

where d is called the order of the tensor [15]. In this paper, all graphically represented tensors follow the convention of drawing tensors introduced in [16].

3.1 Contracting

A tensor of order d can be graphically represented by drawing one “leg” for each dimension like in Figure 4. Multiplying tensors can then be achieved by connecting legs and contracting the resulting tensor network, equal to a summation over the respective tensor indices, while multiplying elements with the same index value [15]. As an example, for $A \in \mathbb{C}^{m \times n}$ and $v \in \mathbb{C}^n$, a matrix-vector multiplication equivalent to

$$Av = \sum_{j=1}^n a_{i,j} v_j \quad (14)$$

can be graphically represented as shown in Figure 5 [14]. The index j runs over both a and v in Equation (14), which was signalled by the contracted leg in Figure 5 connecting a and v .

3.2 Reshaping

A tensor can be reshaped to change its order. Reshaping a tensor does not change its data, instead the resulting new tensor has to be interpreted differently. A state vector $|\psi\rangle \in \mathbb{C}^{2^N}$, representing a system of N sites with basis states $|0\rangle$ and $|1\rangle$, is a tensor of order 1. By reshaping $|\psi\rangle$ into a tensor $|\phi\rangle \in \mathbb{C}^{2 \times 2^{N-1}}$, a new tensor with order 2 is created [14]. The left leg of $|\phi\rangle$ represents the system of the leftmost site of $|\psi\rangle$, while the right leg represents the rest of the system. $|\psi\rangle$ can be recovered by reinterpreting $|\phi\rangle$ as

$$|\phi\rangle_{i,j} = |\psi\rangle_{(i * 2^{N-1} + j)}. \quad (15)$$

Similarly to increasing the order this way, multiple legs of a tensor can be merged to one, thus reducing the tensor’s order [14].

Tensors follow the same principles as flat arrays, representing multi-dimensional data. Increasing the order of a tensor is equivalent to splitting its data into equally sized, continuous chunks of data, but not changing the order. Similarly, decreasing the order is equivalent to flattening a multi-dimensional array.

3.3 Splitting

Tensors can be split into multiple connected tensors forming a tensor network, such that the single tensor and the network represent the same data. To split

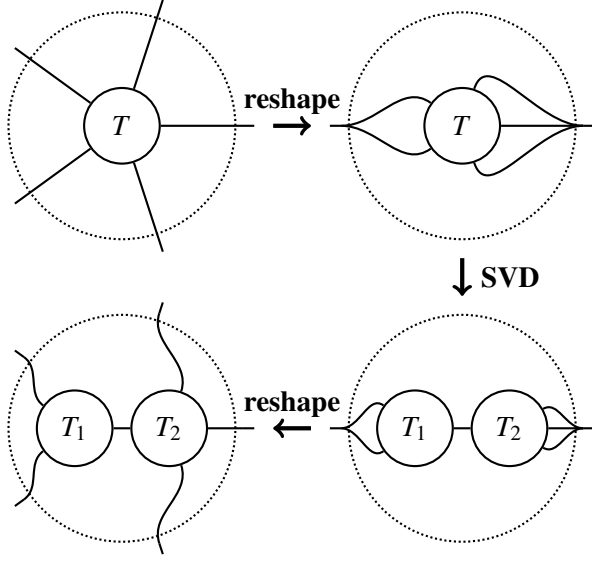


Figure 6 A splitting operation, shown as a sequence of first reshaping a tensor into a tensor of order 2, splitting the matrix via SVD, and finally reshaping the resulting tensors to match the shape of the original tensor. From outside of the dotted circle, the original tensor and the network are equivalent.

a tensor, it first has to be reshaped into a tensor of order 2. The splitting is then performed by matrix decomposition operations like the singular value decomposition (SVD) [17]. The singular values can be truncated or padded to increase the dimension of its connecting legs, before absorbing them into either of the two isometric matrices [18]. After the split, the resulting matrices have to be reshaped to match the shape of the tensor network to the shape of the original tensor. The process is graphically represented in Figure 6. The original tensor can be recovered by contracting the network.

3.4 Matrix Product States

A state vector $|\psi\rangle \in \mathbb{C}^{2^N}$ representing a system of N sites with basis states $|0\rangle$ and $|1\rangle$ can be graphically represented as a tensor of order 1. By repeatedly applying the reshape and split operations, similar to Figure 6, a tensor network of N connected tensors can be created, with each tensor having one open leg of dimension 2. The result is called a matrix product state (MPS) [19], graphically represented for $N = 5$ in Figure 7a. By convention, a MPS with upward-facing legs are the conjugate transpose of the same MPS with downward-facing legs [15].

The connecting legs of the MPS are called bonds, and allow for entanglement between sites [17]. The size of the bond dimensions can be controlled by trun-

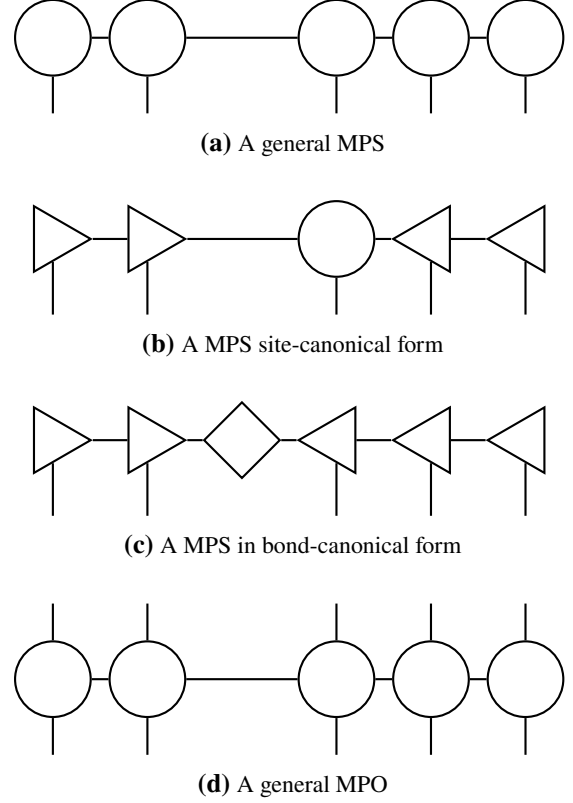


Figure 7 A MPS in different canonical forms (a,b,c) and a MPO (d)

cating or padding the singular values after a SVD during the splitting of a tensor, as described in Section 3.3. If all bonds only have one non-zero singular value, the state $|\psi\rangle$ represented by the MPS is not entangled and is called a product state [18]. Product states can be expressed with the Kronecker product

$$\bigotimes_{i=1}^N s_i, \quad (16)$$

where s_i are the site-states. This makes product states especially efficiently representable in MPS form, by setting $A_i = s_i$ for $i \in \{1, \dots, N\}$ and adding bonds of dimension 1 to connect the tensors. For high values of N , efficient MPS construction methods like this are necessary to avoid ever creating a full state vector representation, which would not be feasible because of its exponential growth.

For a fixed state, the size of a state vectors is constant, whereas the size of a MPS depends on the measure of entanglement and therefore the bond dimensions. If the sites of $|\psi\rangle$ are not entangled, the size of a MPS describing $|\psi\rangle$ grows only linearly in N .

For entangled states, the bond dimensions of a MPS has to be higher than 1 to accurately represent the state [18]. Despite this, the total size of the MPS

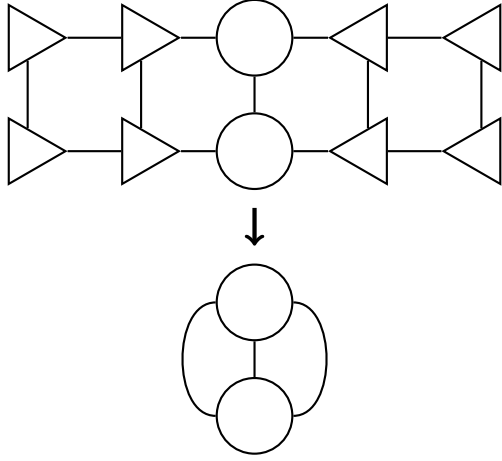


Figure 8 Multiplying a MPS in canonical form with its conjugate transpose simplifies to a single tensor contraction.

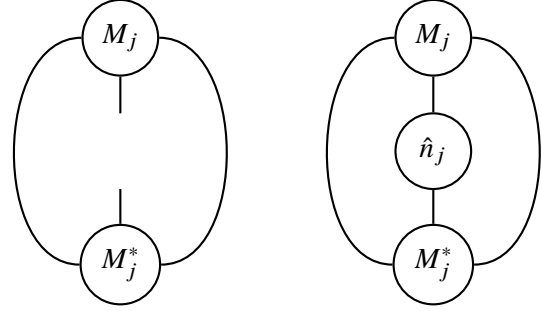
is often still much less than the size of the full state vector, depending on the measure of entanglement. The space requirement can be lowered even further by truncating bond dimensions to values lower than necessary to exactly represent the state, dramatically reducing the size of the MPS, especially if errors are acceptable [18].

3.5 Canonical Forms

After a SVD, the resulting singular values can be absorbed into either of the two resulting isometric matrices, while the other matrix keeps its isometric properties [18]. In a MPS, isometric tensors are called left orthonormal or right orthonormal, and are graphically represented by drawing a triangle instead of a circle [18]. Following this, a MPS is in site-canonical form if there is a tensor, called the orthogonality center, such that all other tensors are in orthonormal form, like in Figure 7b [18].

The singular values do not have to be absorbed into either tensor, and can instead be kept as a part of the MPS. The result is a MPS with an additional bond tensor of order 2 connecting two site tensors, graphically represented as a diamond [18]. A MPS is in bond-canonical form if there is a bond tensor, called the orthogonality center, such that all other tensors the MPS consists of are site tensors in orthonormal form, shown in Figure 7c [18].

By virtue of the orthonormality of its tensors, a MPS in canonical form can be efficiently multiplied with its conjugate transpose. The multiplication can be graphically represented like in Figure 8, where the tensor net can be reduced to a single tensor contraction [18].



(a) ρ_j : Partial trace of site j (b) Measurement using ρ_j

Figure 9 Tensor networks illustrating the partial trace of site j of a state and how it can be used to measure an observable \hat{n}_j at site j

3.6 Measurement & SSE

Expressing the site-measurement from Equations (3) and (4) in the MPS framework results in Figure 9b. The observable \hat{n}_j is contracted with a MPS in site-canonical form and its conjugate transpose, the orthonormal tensors simplify to identities and the resulting tensor net can be contracted efficiently.

By omitting the tensor \hat{n}_j and leaving the legs connecting to it open, the resulting tensor network represents the density matrix ρ_j of the partial trace of site j . ρ_j can be used for the measurement of any observable on site j or interpreted as a matrix to compute the single-site entropy (SSE) of site j :

$$S(\rho_j) = -\text{Tr}(\rho_j \log_2(\rho_j)). \quad (17)$$

$S(\rho_j)$ provides a measure of entanglement between site j and the rest of the system. Its values range from 0 (no entanglement) to 1 (maximal entanglement) [12].

3.7 Matrix Product Operators

Analogously to a MPS describing a state vector, a matrix product operator (MPO) is a tensor network describing an operator. While a MPS consists of tensors of order 3, a MPO consists of tensors of order 4 having an additional free leg, as shown in Figure 7d [18].

Given some operator \hat{H} in matrix form, a MPO representing the same operator can be constructed by repeated reshaping and splitting operations, similarly to constructing MPSs. However, just like with state vectors, the size of \hat{H} grows exponentially with N . Therefore, the MPO has to be constructed without constructing \hat{H} in matrix form. For the case like this, where Ising Hamiltonians are used, this can be done by using state automata to construct an Ising-Hamiltonian in MPO-form [20, 21].

4 TDVP

The TDVP reduces the time evolution of a complete MPS to evolutions per site and per bond. Each site or bond tensor is evolved one after the other, by sweeping back and forth through the MPS in a Trotter scheme [22]. The bond dimensions are often constrained to some upper limit by projecting the time evolution into a manifold of MPSs of some initial bond dimension [22].

4.1 1TDVP

A single-site variation of a state in MPS form $|\psi\rangle$ is another MPS obtained by changing a single-site tensor of $|\psi\rangle$ [23]. Furthermore, the single-site tangent space $T_{|\psi\rangle}$ is a space spanned by all single-site variations of $|\psi\rangle$. The projector $\hat{P}_{T_{|\psi\rangle}}$ which projects on this tangent space is given by

$$\hat{P}_{T_{|\psi\rangle}} = \sum_{j=1}^L \hat{P}_{j-1}^{L,|\psi\rangle} \otimes \hat{\mathbf{1}} \otimes \hat{P}_{j+1}^{R,|\psi\rangle} - \sum_{j=1}^{L-1} \hat{P}_j^{L,|\psi\rangle} \otimes \hat{P}_{j+1}^{R,|\psi\rangle}, \quad (18)$$

where $\hat{P}_j^{L,|\psi\rangle}$ and $\hat{P}_j^{R,|\psi\rangle}$ are projectors onto the sites left or right of and including site j of $|\psi\rangle$ [23]. “The first contributing sum of Equation (18) filters for all MPSs which differ at most on one site from $|\psi\rangle$ ” [23], represented as a tensor net for $j = 4$ in Figure 10. The second contributing sum removes all states which are equivalent to $|\psi\rangle$ [23].

To arrive at the desired Trotter splitting, $\hat{P}_{T_{|\psi\rangle}}$ is used to project the Hamiltonian into the single-site tangent space and mapping the resulting SE to the corresponding site or bond [23]. The result is a set of new SEs, a forwards evolution for each site and a backwards evolution for each bond:

$$\frac{\partial}{\partial t} M_j = -i\hat{H}_j^{\text{eff}} M_j \quad (19)$$

$$\frac{\partial}{\partial t} C_{\underline{j}} = +i\hat{H}_{\underline{j}}^{\text{eff}} C_{\underline{j}}. \quad (20)$$

\hat{H}_j^{eff} and $\hat{H}_{\underline{j}}^{\text{eff}}$ are the effective Hamiltonians for site j and bond \underline{j} , respectively [23], shown in Figures 11a and 11b.

Finally, the sites and bonds are updated in an order to facilitate sweeping back and forth through the MPS, updating site and bond tensors with the solutions from the SEs (19) and (20) [23].

The measure of entanglement between sites often fluctuates during a simulation, requiring a growth or reduction of the bond dimensions to avoid errors or redundancies. However, because of the projection of the Hamiltonian in this 1-site TDVP (1TDVP) and the time evolution being constrained to the subspace of MPSs of the chosen initial bond dimension, projection errors are hard to predict. Therefore, the bond dimensions of the MPS are kept constant, so the dimension of the initial state must be padded to be high enough to avoid errors during the simulation.

The time needed to solve the SEs is in $O(n^3)$ [24], quickly growing with the size n of the combined tensor. Consequently, unnecessarily high bond dimensions, and therefore larger than needed tensors should be avoided. At the same time, bond dimensions must be large enough for the MPS to be able to accurately represent the state. Ideally the bond dimension should be adjusted over time to match a value dictated by the measure of entanglement and some error tolerance. Methods for doing this will be discussed in the following.

4.2 2TDVP

The 1TDVP can be extended from a single site variant to the 2-site TDVP (2TDVP) [16]. This is achieved by performing time evolutions on the combined 2-site tensor governed by a 2-site effective Hamiltonian, as shown in Figure 11c [23]. Two neighboring tensors have to be contracted before evolving the combined tensor, and split afterwards. Subsequently, the same backwards bond evolution Equation (20) as for the 1TDVP is performed for the tensor further along the sweep direction [23]. The order of evolutions follows a back and forth sweeping pattern like in the 1TDVP.

An advantage of the 2TDVP is the possibility to change the bond dimensions of the MPS on the fly [16]. The forwards evolution is done on a combined tensor, enabling entanglements between both sites without the constraint of a bond leg. After splitting a 2-site tensor, the dimension of the bond leg connecting the two resulting tensors can be adjusted by inspecting the singular values, as described in Section 3.3. This way, the bond dimensions of the MPS can grow to accommodate the new state, depending on a chosen error tolerance. However, in cases where the bond dimension reaches its peak, this flexibility comes at the cost of having to solve a SE in the higher dimension of a combined 2-site tensor, compared to the 1TDVP.

Whether the 2TDVP is faster than the 1TDVP for a given simulation depends on how much the required

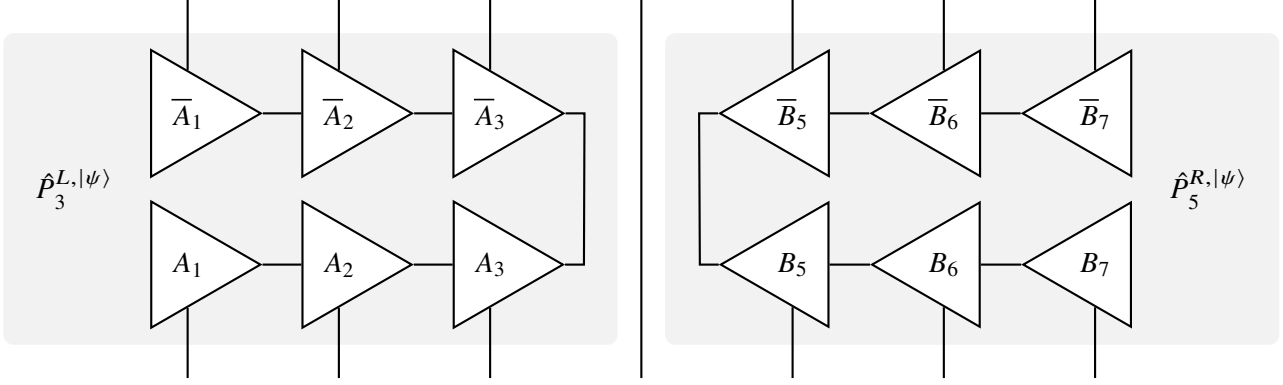


Figure 10 $\hat{P}_{T|\psi}$: A tensor net filtering for all MPSs which may differ from $|\psi\rangle$ only on site 4, equivalent to $\hat{P}_3^{L,|\psi\rangle} \otimes \hat{\mathbf{1}} \otimes \hat{P}_5^{R,|\psi\rangle}$, where A_j/B_j and \bar{A}_j/\bar{B}_j are the left orthonormal and right orthonormal site tensors of $|\psi\rangle$ and their complex conjugates, respectively

bond dimension fluctuates and whether or not a good estimate for an initial bond dimension in the 1TDVP is available.

4.3 A1TDVP

One variant of the TDVP with dynamically evolving bond dimensions is the adaptable 1TDVP (A1TDVP) [19]. The algorithm aims to modify the 1TDVP algorithm with the ability to change the bond dimensions of the MPS on the fly, without having to work with the expensive 2-site tensors from the 2TDVP. However, choosing to dispense with combined tensors means the required bond dimensions have to be estimated in a different way.

In the A1TDVP, this is done by calculating a convergence measure for each site i , depending on its dimension d [19]:

$$f(i)_d = \|\hat{H}_i^{\text{eff}} M_i\|^2 + \|\hat{H}_i^{\text{eff}} C_i\|^2 + \|\hat{H}_{i+1}^{\text{eff}} M_{i+1}\|^2, \quad (21)$$

with \hat{H}_i^{eff} and \hat{H}_i^{eff} the effective site and bond Hamiltonians from Equations (19) and (20). The dimension of bond i is then increased and $f(i)$ is recalculated. Convergence is reached when the quotient of the new convergence measure and its previous convergence measure are smaller than some error tolerance p [19]:

$$\frac{f(i)_{d+1}}{f(i)_d} - 1 \leq p. \quad (22)$$

“The absolute value of $f(\tilde{D}_i)$ has no meaning” [19], however, its convergence indicates that increasing d will have a negligible effect on the projection error. The new bond dimensions are chosen to be the smallest d for which Equation (22) holds [19].

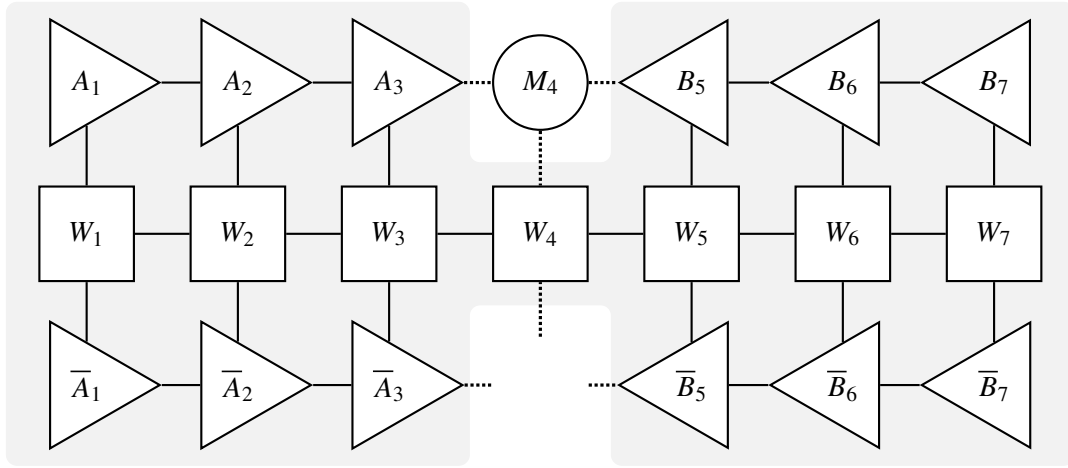
Due to variable bond dimensions, while also preventing large combined site tensors, the A1TDVP is

more efficient than the 2TDVP in some scenarios. However, determining new bond dimensions works very differently in both algorithms which complicates comparisons between the two. The only tweaking parameter, error tolerance, has a different meaning in both algorithms. In order to make a comparison between the algorithms, a pair of tolerances has to be found that produces a similar bond dimension growth pattern in both algorithms. In my testing however, most often the bond dimensions in the A1TDVP seemed to grow more quickly and to higher values than in the 2TDVP, regardless of the error tolerance. This in turn produced larger tensors and increased the time needed to solve the respective SEs.

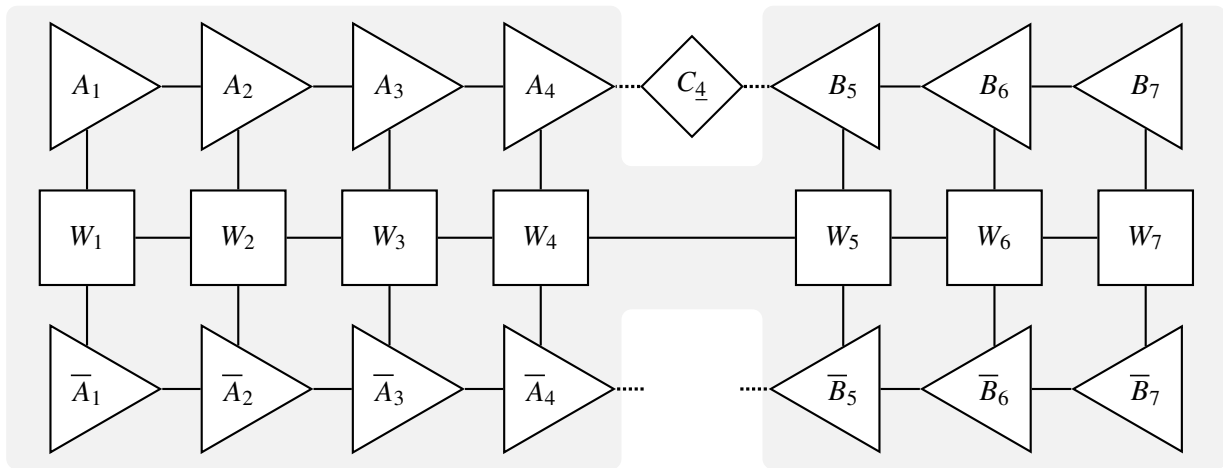
5 New A1TDVP and Results

Faced with the potentially high efficiency of the A1TDVP, but also the difficulty to predict when it will produce good results, I implemented a new variant of the TDVP. The n1TDVP combines the idea of calculating new bond dimensions between sweeps, like in the A1TDVP, with the accuracy of determining new bond dimensions via the SVD like in the 2TDVP.

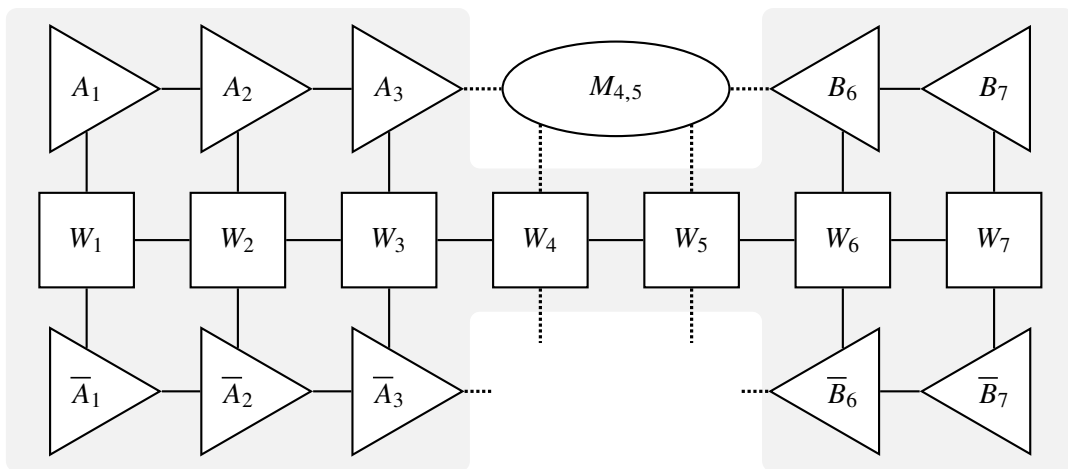
The key to reducing calculation time is to avoid solving SEs for combined tensors, because of the rapidly growing time requirement to do so for large tensors. However, if a time evolution is only performed on single-site tensors, no SVD can be done to decide whether to change the bond dimensions. It turns out that approximating the effect of an exact time evolution on a combined tensor is much faster than solving the respective SE, while still providing insight into the measure of entanglement. In the n1TDVP, this is done for each bond by contracting its neighboring tensors, per-



(a) $\hat{H}_4^{\text{eff}} \cdot M_4$



(b) $\hat{H}_4^{\text{eff}} \cdot M_4$



(c) $\hat{H}_{4,5}^{\text{eff}} \cdot M_{4,5}$

Figure 11 Tensor networks representing the product of a tensor and an effective Hamiltonian governing the evolution of a single site (a), a bond (b), or two combined sites (c) in the TDVP algorithm

forming some approximative evolution method, splitting the tensor via a SVD, and deciding a new bond dimension by inspecting the singular values. The evolution method can be anything that approximates an exact time evolution. The Taylor expansion of the matrix exponential is an obvious choice, as the number of steps can be just high enough to provide a good approximation, while still being time efficient.

5.1 Results

Figures 12b to 12d show plots done with the n1TDVP for different amounts of Taylor steps and their respective running times. Compared with the 2TDVP, shown in Figure 12a, the n1TDVP is faster in every case, while producing almost identical plots. The result also indicates that 1 Taylor step is already enough to produce a good approximation, speeding up the simulation considerably.

The growth of the bond dimensions is similar for all methods, growing slightly slower for the 2TDVP case. The 2TDVP serves as a reference of how the bond dimension should grow according to the chosen error tolerance. The slightly higher bond dimensions in the n1TDVP slow down the calculation while providing a negligible increase in accuracy. The difference in bond dimension growth is expected due to the inaccuracies in the Taylor method for approximating a time evolution. However, as long as the bond dimensions are not lower than in the 2TDVP case, the simulation will not be less accurate than dictated by the error tolerance.

Unfortunately, the n1TDVP is not strictly faster than other variants of the TDVP algorithm. Similar to the A1TDVP, for some configurations the bond dimensions grow a lot faster than in the 2TDVP. In some cases where the highest bond dimension needed can be approximated beforehand, the issue can be alleviated by limiting the bond dimension, preventing uncontrolled growth.

5.2 Conclusion and Future Work

The n1TDVP is yet another spin on the TDVP algorithm with much more left to explore. For simulating QCAs, the algorithm shows potential in some scenarios, outpacing the known 2TDVP algorithm without sacrificing accuracy. In other scenarios however, it grows the MPS bond dimension to larger-than-needed values, slowing down dramatically. Future work could therefore consist of using approximative evolution methods that show a more accurate behavior than the

Taylor series, with one potential candidate being the Krylov time evolution.

Faster MPS time evolution methods in turn enable deeper investigation of QCAs. Regarding other areas, future research could study the effect of changing the observable for site measurements in Equation (4), or explore periodic behavior for different automata rules and cell numbers. Furthermore, the n1TDVP is not limited to QCAs and could be tested for other MPS-based simulations, unrelated to automata.

Glossary

1TDVP 1-site TDVP.

2TDVP 2-site TDVP.

A1TDVP adaptable 1TDVP.

CA cellular automaton.

MPO matrix product operator.

MPS matrix product state.

n1TDVP new 1TDVP.

QCA quantum cellular automaton.

SE Schrödinger equation.

SSE single-site entropy.

SVD singular value decomposition.

TDVP time-dependent variational principle.

References

- [1] Marco Tomassini, Moshe Sipper, and Mathieu Perrenoud. “On the generation of high-quality random numbers by two-dimensional cellular automata”. In: *IEEE Transactions on computers* 49.10 (2000), pp. 1146–1151. DOI: 10.1109/12.888056.
- [2] Dieter A Wolf-Gladrow. *Lattice-gas cellular automata and lattice Boltzmann models: an introduction*. Springer, 2004. DOI: 10.1007/b72010.

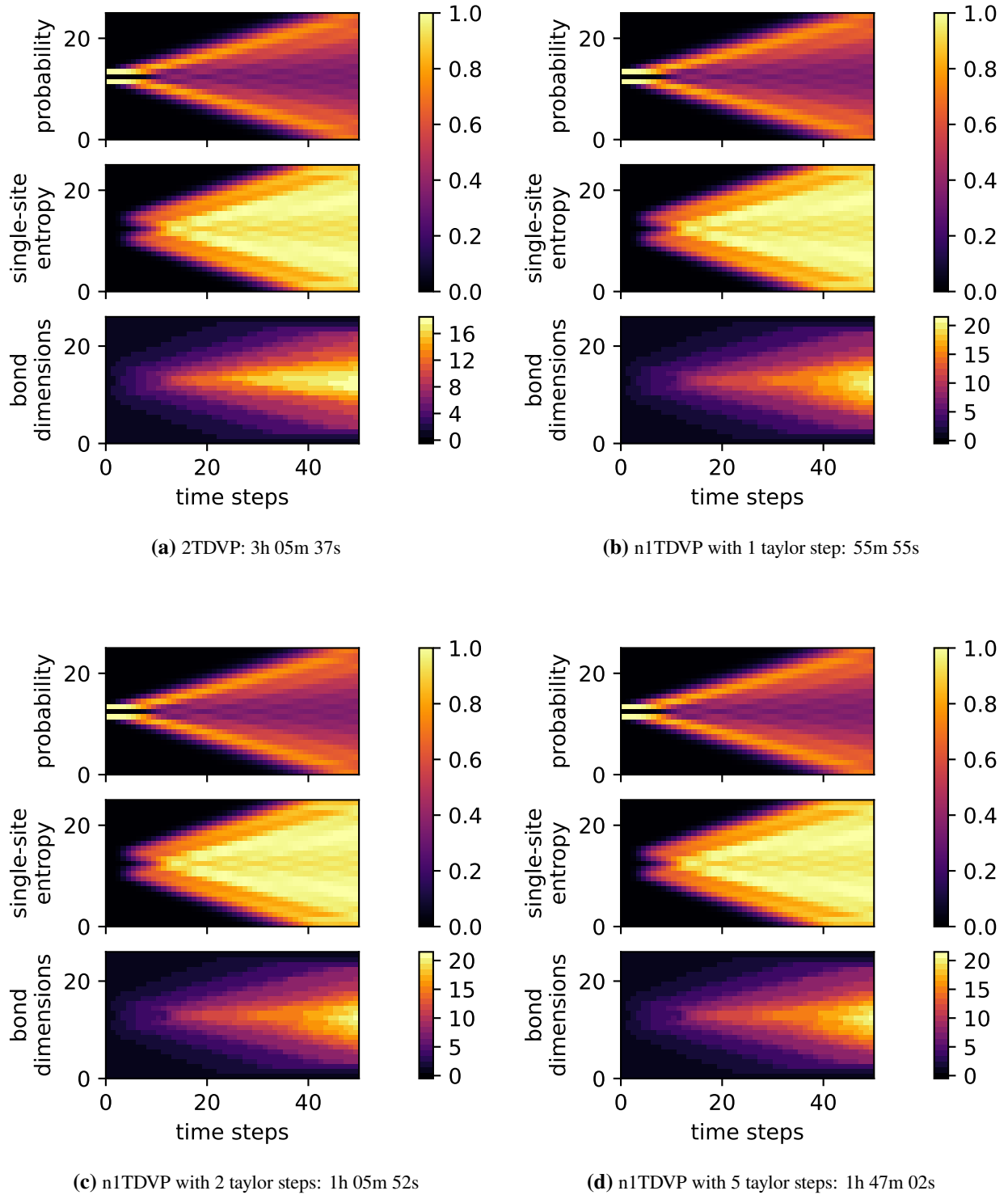


Figure 12 Time evolution of the initial state $|\psi\rangle_0 = |0\rangle^{11} \otimes |101\rangle \otimes |0\rangle^{11}$ governed by the rule: “A cell is flipped iff the number of alive cells among its nearest neighbors is exactly 1”. A time step is plotted after every $t_{\text{plot}} = \frac{\pi}{20}$ interval, with a step size of $t = \frac{\pi}{400}$ being used in the calculation. The evolution is plotted using different algorithms, with their respective running times annotated below. Simulations each ran on one Node of the LRZ-Linux Cluster CoolMUC-2, having 28 cores, 2 hyperthreads per core, 2.6 GHz core nominal frequency, and 120 GB/s memory bandwidth. The implementation used is available on GitHub [7].

- [3] M. Gerhardt and H. Schuster. “A cellular automaton describing the formation of spatially ordered structures in chemical systems”. In: *Physica D: Nonlinear Phenomena* 36.3 (1989), pp. 209–221. ISSN: 0167-2789. DOI: [https://doi.org/10.1016/0167-2789\(89\)90081-X](https://doi.org/10.1016/0167-2789(89)90081-X).
- [4] D. Bleh, T. Calarco, and S. Montangero. *Quantum Game of Life*. 2012. DOI: 10.48550/arXiv.1010.4666. arXiv: 1010.4666 [quant-ph].
- [5] Terry Farrelly. “A review of Quantum Cellular Automata”. In: *Quantum* 4 (2020-11), p. 368. ISSN: 2521-327X. DOI: 10.22331/q-2020-11-30-368.
- [6] John Preskill. “Quantum Computing in the NISQ era and beyond”. In: *Quantum* 2 (2018), p. 79. DOI: 10.22331/q-2018-08-06-79.
- [7] Benjamin Decker. *quantum-cellular-automaton*. URL: <https://github.com/BenjaminDecker/quantum-cellular-automaton>.
- [8] Martin Gardner. “Mathematical Games: The Fantastic Combinations of John Conway’s New Solitaire Game “Life””. In: *Sc. Am.* 223 (1970), pp. 120–123. DOI: 10.1038/scientificamerican1070-120.
- [9] Elwyn R Berlekamp, John H Conway, and Richard K Guy. *Winning Ways for Your Mathematical Plays, Volume 3*. CRC Press, 2018. DOI: 10.1201/9780429487316.
- [10] Johnny Nyquist. *Ca110-structures2.png*. 2012. URL: <https://commons.wikimedia.org/w/index.php?curid=18642040>.
- [11] Eytan Domany and Wolfgang Kinzel. “Equivalence of Cellular Automata to Ising Models and Directed Percolation”. In: *Phys. Rev. Lett.* 53 (4 1984), pp. 311–314. DOI: 10.1103/PhysRevLett.53.311.
- [12] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.
- [13] Peter-Maximilian Ney et al. “Entanglement in the quantum Game of Life”. In: *Physical Review A* 105.1 (2022), p. 012416. DOI: 10.1103/PhysRevA.105.012416.
- [14] Univ.-Prof. Dr. Christian Mendl. *Lecture: Tensor Networks, Technical University Munich IN2388*. 2022.
- [15] Jacob Biamonte. *Lectures on Quantum Tensor Networks*. 2020. DOI: 10.48550/arXiv.1912.10049. arXiv: 1912.10049 [quant-ph].
- [16] Jutho Haegeman et al. “Unifying time evolution and optimization with matrix product states”. In: *Physical Review B* 94.16 (2016). DOI: 10.1103/physrevb.94.165116.
- [17] Jacob C Bridgeman and Christopher T Chubb. “Hand-waving and interpretive dance: an introductory course on tensor networks”. In: *Journal of physics A: Mathematical and theoretical* 50.22 (2017), p. 223001. DOI: 10.1088/1751-8121/aa6dc3.
- [18] Ulrich Schollwöck. “The density-matrix renormalization group in the age of matrix product states”. In: *Annals of Physics* 326.1 (2011), pp. 96–192. DOI: 10.1016/j.aop.2010.09.012.
- [19] Angus J Dunnett and Alex W Chin. “Dynamically Evolving Bond-Dimensions within the one-site Time-Dependent-Variational-Principle method for Matrix Product States: Towards efficient simulation of non-equilibrium open quantum dynamics”. In: *arXiv preprint arXiv:2007.13528* (2020). DOI: 10.48550/arXiv.2007.13528.
- [20] Gregory M. Crosswhite and Dave Bacon. “Finite automata for caching in matrix product algorithms”. In: *Physical Review A* 78.1 (2008). DOI: 10.1103/physreva.78.012356.
- [21] F. Fröwis, V. Nebendahl, and W. Dür. “Tensor operators: Constructions and applications for long-range interaction systems”. In: *Physical Review A* 81.6 (2010). DOI: 10.1103/physreva.81.062337.
- [22] Jutho Haegeman et al. “Time-Dependent Variational Principle for Quantum Lattices”. In: *Physical Review Letters* 107.7 (2011). DOI: 10.1103/physrevlett.107.070601.
- [23] TensorNetwork.org contributors. *The time-dependent variational principle*. URL: <https://tensornetwork.org/mps/algorithms/timeevo/tdvp.html> (visited on 2023-04-25).
- [24] Cleve Moler and Charles Van Loan. “Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later”. In: *SIAM review* 45.1 (2003), pp. 3–49. DOI: 10.1137/S00361445024180.