CoTenN: Constrained Optimization with Tensor Networks

RITVIK SHARMA, Stanford University, USA
CHENG PENG, SLAC National Accelerator Laboratory, USA
SIDDHARTH DANGWAL, University of Chicago, USA
SARA ACHOUR, Stanford University, USA

Simulation of physics problems is one of the most important use cases of quantum computing. For this class of problems, the goal is typically to find the minimum energy state, or the ground state, of a physical system's Hamiltonian. These problems frequently have constraints, such as symmetry conditions, which must also be satisfied. To solve such problems, researchers in computational physics use quantum-inspired algorithms that execute on classical computers. In particular, tensor network-based eigensolvers such as DMRG have become popular.

However, to use these eigensolvers, the constrained optimization problem must first be encoded as a tensor network that implements a low-rank decomposition of the system's Hamiltonian and state vector. These tensor network encodings are highly flexible, allowing for variables with ≥ 2 quantum states and supporting efficient constraint encodings that directly constrain the state vector. A critical challenge to developing tensor network-based encodings is that, currently, the encoding process is manual; significant effort is required to identify an efficient encoding for a new physics problem. In this work, we introduce a quantum constrained optimization problem (QCOP), a general abstraction for describing minimization problems over quantum variables that are subject to hard constraints. We present Masq, the first constraint programming language for QCOPs implementable with tensor networks, and CoTenN, a compiler that automatically maps QCOPs specified with Masq programs to tensor networks. To demonstrate the utility of Masq and CoTenN, we formulate two physics problems in Masq and then use CoTenN to find their ground states. We find the CoTenN-generated tensor networks generally outperform SOTA problem formulations, providing between 2.05×–53.32× total speedups across runs for QCOPs and yielding up to $2.49 \cdot 10^7\times$ lower truncation errors for otherwise unconstrained problems.

CCS Concepts: • **Computer systems organization → Quantum computing**.

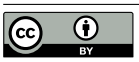
Additional Key Words and Phrases: Quantum Simulation, Tensor Networks, Optimization Problems

ACM Reference Format:
Ritvik Sharma, Cheng Peng, Siddharth Dangwal, and Sara Achour. 2026. CoTenN: Constrained Optimization with Tensor Networks. *Proc. ACM Program. Lang.* 10, PLDI, Article 194 (June 2026), 27 pages. https://doi.org/10.1145/3808272

1 Introduction

Quantum as a computing paradigm was first proposed to simulate realistic quantum many-body physics and quantum chemistry problems by Richard Feynman [15]. Such simulations are computationally hard, but enable us to determine important physical quantities like the bond energy, activation energy, etc., for physical systems [3, 11, 37, 49, 66]. A typical setting for such problems involves defining a *Hamiltonian* and finding the minimum energy state, or the *ground state*, of this Hamiltonian. For many of these problems, the ground state is itself a superposition of states and therefore probabilistic. The system states are subject to constraints arising from symmetries of the

Authors' Contact Information: Ritvik Sharma, Stanford University, Stanford, USA, rsharma3@stanford.edu; Cheng Peng, SLAC National Accelerator Laboratory, Menlo Park, USA, cpeng18@slac.stanford.edu; Siddharth Dangwal, University of Chicago, Chicago, USA, siddharthdangwal@uchicago.edu; Sara Achour, Stanford University, Stanford, USA, sachour@stanford.edu.



This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 2475-1421/2026/6-ART194
https://doi.org/10.1145/3808272

system [52]. Performing an exact simulation of these systems on classical computers comes with exponential overheads. For example, a molecule with N electrons requires $O(2^N)$ floating point numbers just to store its quantum state exactly. In this setting, simulating Chromium Dimer (often considered a benchmark for quantum simulation methods) [13, 44, 60], would require 256 Terabytes of memory, putting it beyond the reach of most classical systems.

To simulate these systems approximately, researchers have developed quantum-inspired classical algorithms that run on commodity hardware [17, 62, 63]. Among these, tensor network (TN) methods have become widely used for performing approximate Hamiltonian simulations for problems in combinatorial optimization [2, 23, 54], quantum chemistry [10, 28, 39], electrodynamics [20, 34], and differential equations [58]. TNs are low-rank approximations of quantum systems that grow polynomially rather than exponentially with problem size, enabling compact storage on digital hardware. TN algorithms like DMRG perform numeric computation on TNs without fully expanding them, circumventing any exponential blow-up in memory usage and making large-scale quantum simulations feasible [17, 55, 63, 64]. Advances in TN methods have also enabled study of increasingly complex quantum problems with classical resources accessible today, informing the design and theory of future quantum computing devices.

Challenges with Encoding Constraints. Physically realistic Hamiltonians often have conservation laws and symmetries that partition the Hilbert space into disjoint sectors, constraining the state of the system. Respecting these physical constraints improves simulation performance and ensures correct behavior. Regularization-based methods, which add penalty terms into the Hamiltonian to capture constraints, are popularly used since they are easy to implement [5–7, 12, 22]. However, they fail to efficiently eliminate invalid states from the search space, resulting in longer convergence times and increased simulation error.

To improve efficiency and accuracy, domain experts often manually craft bespoke TN representations for specific problems that embed constraints directly. Developing such encodings requires considerable expertise and creativity, to the extent where publications typically focus on developing a TN implementation for a single problem [9, 23, 36, 39, 43, 58]. Practitioners use TN libraries such as ITensors, cuTensorNet, and TenPy to formulate these encodings [14, 21, 27, 35, 46]. Because these encodings (summarized in Table 8) are highly specialized, they find little reuse across models and their constraint structures cannot be readily combined.

1.1 Specifying and Simulating QCOPs with CoTenN

Despite the ubiquity of such Hamiltonian simulations and constraints, no general abstraction exists for expressing and enforcing them. In this work, we introduce the quantum constrained optimization problem (QCOP), a problem abstraction that supports specification of physical constraints along with the Hamiltonian of the system. We introduce a programming framework (Masq) for expressing QCOPs that can be efficiently and automatically mapped to a TN. Masq is the first constraint programming language for QCOPs. We also present the compiler CoTenN, that automatically generates a TN that implements the described problem and dispatches it to a backend eigensolver. CoTenN and Masq offer the following benefits:

- *Separation of Concerns.* The QCOP abstraction and the Masq language enable separation of concerns: users can specify the problem and its constraints without providing the TN implementation.
- *Ergonomic QCOP design.* Masq provides language constructs for specifying QCOP Hamiltonians and constraints without requiring users to reason about constraint mappings. This eliminates problem-specific manual derivation, enables constraint reuse, and supports composition of constraints.
- *Intelligent TN constraint encodings.* CoTenN efficiently maps Masq constraints to a TN. This mapping was developed by generalizing various bespoke constraint encodings found in literature.

To our knowledge, CoTenN is the first concerted attempt to generalize bespoke TN encoding strategies so that they may be more broadly used, and the first attempt at devising a programming abstraction that separates the specification of physics problems and their constraints from implementation.

1.2 Contributions

- We introduce the idea of a constrained quantum optimization problem (QCOP), and provide a primer for how QCOPs are encoded with TN. We present Masq, a language for describing QCOPs that can be automatically mapped to TN. We formulate two physics benchmarks, the Kitaev Ladder and the PXP model, using Masq to demonstrate its flexibility.
- We present CoTenN, a compiler that translates QCOP problems to TNs and systematically applies direct constraint encoding techniques to effectively impose hard constraints on the system.
- We present a collection of generalized constraint encodings developed by analyzing bespoke constraint encodings from TN literature. We also present a new TN encoding for eigenvalue constraints, which can be used to encode Kitaev ladder model benchmark [10].
- We evaluate CoTenN-generated TN implementations against regularization-based formulations. We find that CoTenN-generated TNs provide between 2.05×–53.32× total speedups across runs for QCOPs and yield up to $2.49 \cdot 10^7\times$ lower truncation errors for otherwise unconstrained problems.

2 Background with Motivating Example

In this section, we describe quantum optimization problems (QOPs) and constrained quantum optimization problems (QCOPs), introduce an example problem, and motivate how constraints can be used for such problems.

2.1 Quantum Optimization Problems (QOP)

A QOP is the problem of finding the lowest energy state, or ground state, of a quantum system.

▸ *Quantum Variables.* The domain of the optimization problem is defined over N discrete quantum variables $x_1,...,x_N$, where each variable exists in a superposition of D orthogonal values given by $\{0,...,D-1\}$. The values of the variables are interdependent, or entangled, and so the system exists in a superposition of D^N quantum states. Each quantum state maps to a combination of variable values.

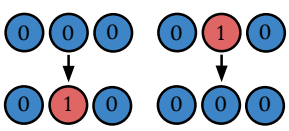
The state can be formally described with a D^N-dimensional complex-valued vector $\vec{s} \in \mathbb{C}^{D^N}$. Each entry $s[\mathbf{j}]$ corresponds to a state configuration $\mathbf{j} = (j_1,...,j_N) \in \{0,...D\}^N$ where x_i is assigned state j_i. Each element in this vector, $\vec{s}_i[\mathbf{j}] = a_i[\mathbf{j}] \cdot e^{\mathrm{i} \cdot b_i[\mathbf{j}]}$, captures the amplitude of x_i being in state $j \in \{0,...,D-1\}$ upon measurement, with measurement probability $P(x_i = j) = |a_i[j]|^2$.

▸ *Objective Function.* Given a set of N quantum variables, a QOP is defined by a Hamiltonian matrix $H \in \mathbb{C}^{D^N \times D^N}$ that encodes interactions between quantum states. The goal is to find the normalized state vector $\vec{s} \in \mathbb{C}^{D^N}$ that minimizes the energy:

$$\vec{s}_{\text{opt}} = \underset{\vec{s}}{\operatorname{argmin}}\, E(\vec{s}) \quad \text{where} \quad E(\vec{s}) = \sum_{\mathbf{i},\mathbf{j}} H[\mathbf{i},\mathbf{j}]\, s[\mathbf{i}]\, s[\mathbf{j}]^* = \vec{s}^{\dagger} H \vec{s}$$

The solution, $\vec{s}_{\text{opt}}$, is the superposition of quantum states that has the lowest energy. It is the eigenvector of H corresponding to its smallest eigenvalue. This eigenvalue is the ground-state energy.

▸ *Classical vs Quantum.* For a classical system, H is diagonal and this problem reduces to selecting a position in the diagonal H with the minimum value. However, in general, for non-diagonal Hamiltonians, $\vec{s}_{\text{opt}}$ is a superposition of multiple configurations or orthogonal basis states, each with a nonzero amplitude. The square of this amplitude is the probability of measuring the system in that basis state.

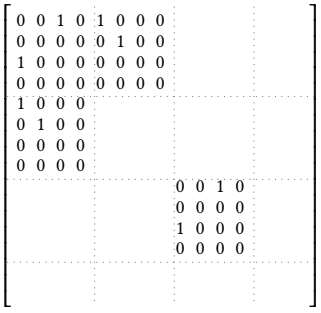


(a) Color transitions

$$\begin{pmatrix}1&0\\0&0\end{pmatrix}\otimes\begin{pmatrix}0&1\\1&0\end{pmatrix}\otimes\begin{pmatrix}1&0\\0&0\end{pmatrix}$$

Left node is blue | Flip node color | Right node is blue

(b) Encoded Operator



(c) Full 16×16 Hamiltonian. Omitted parts of the matrix are all 0.

$$\vec{s}_{\text{opt}} = \tfrac{1}{\sqrt{2}}|BBBB\rangle - \tfrac{1}{2}|BRBB\rangle - \tfrac{1}{2}|BBRB\rangle$$

(d) Solution State

$$C(\mathbf{i}) = \prod_{j=1}^{3}\left(1-\mathbf{i}_j\cdot\mathbf{i}_{j+1}\right)$$

(e) Constraint Function

$$P = \text{diag}(\overbrace{1,1,1,0}^{|00{**}\rangle}, \overbrace{1,1,0,0}^{|01{**}\rangle}, \overbrace{1,1,1,0}^{|10{**}\rangle}, \overbrace{0,0,0,0}^{|11{**}\rangle})$$

(f) Constraint Projection Operator

Fig. 1. System description, Hamiltonian, solution state and constraints for the Quantum Color Flipping Game.

2.2 Quantum Constrained Optimization Problems (QCOPs)

In a quantum constrained optimization problem (QCOP), the goal is to minimize an energy function by finding the state of a set of quantum variables where there is a hard constraint on allowed states of the variables. The goal is to find $\vec{s}_{\text{opt}}$ that minimizes the energy and satisfies the constraint:

$$E(\vec{s}) = \sum_{\mathbf{i},\mathbf{j}} H[\mathbf{i},\mathbf{j}]s[\mathbf{i}]s[\mathbf{j}]^* = \vec{s}^{\dagger} H \vec{s} \quad \texttt{subject to } \forall_{\mathbf{i}} s[\mathbf{i}] \neq 0 \implies C(\mathbf{i}) = 1$$

Where $C : \{0,\ldots,D-1\}^N \rightarrow \{0,1\}$ is a constraint function that maps a configuration over all quantum variables $\mathbf{i}$ to 1 if it satisfies the constraint and 0 otherwise, thus eliminating invalid states.

2.3 QCOP Example: Quantum Color Flipping Game.

We start with an example QOP inspired by the structure of the PXP model from our benchmarks (Section 6.1.2). Here we have a chain of $N = 4$ nodes, each of which can be associated with a color: blue or red. The colors are mapped to quantum states by assigning blue to $|B\rangle = |0\rangle$ and red to $|R\rangle = |1\rangle$. The system applies the transitions shown in Figure 1a to any three sequential nodes. The transition rules for this game are implemented as an operator that flips a node's color when both its neighbors are blue, shown in Figure 1b. The corresponding full Hamiltonian over all 4 variables is shown in Figure 1c. Figure 1d shows the ground state of the system.

Constraints in the Quantum Flipping Game. Note that while the QOP formulation has quantum states that span the $2^N = 16$-dimensional space, the Hamiltonian shown in Figure 1c is quite sparse. This sparsity arises because the Hamiltonian does not modify any chain configurations which contain nodes colored red-red. States with such configurations, such as $|BBRR\rangle$ and $|RRRB\rangle$, do not participate in the system dynamics and can not contribute to the ground state. We formalize this as a constraint C shown in Figure 1e. This constraint evaluates to 1 when no adjacent pair is $(1,1)$, and 0 otherwise. This can be implemented as a diagonal projection that applies the identity operator to valid states, and 0 to invalid states (defined as P in Figure 1f).

3 Solving QCOPs/QOPs with Tensor Networks (TNs)

The QCOP and QOP problem formulations in Section 2.3 introduce $\mathbb{C}^{D^N}$ state vectors and a $\mathbb{C}^{D^N} \times \mathbb{C}^{D^N}$ Hamiltonian. Thus, to find the eigenvalues and eigenvectors of a Hamiltonian H, one in principle would need to construct the full matrix which grows exponentially with N, quickly making the cost of storing and operating on it intractable. Even if H is sparse or has local structure, explicitly forming

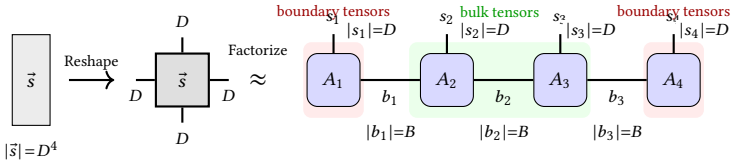


Fig. 2. MPS: Low-Rank Encoding of a state vector $\vec{s} \approx \prod_{k=1}^{N} A_k$. For N quantum variables each with D states, the state vector size is D^N. The MPS tensor A_k stores the state of quantum variable x_k on physical dimension s_k. Tensors A_k,A_{k+1} have a shared dimension b_k called the "bond dimension". Its size (B) captures the number of ranks retained between neighboring A_ks during factorization. The "boundary" tensors A_1,A_4 are $D{\times}B$-dimensional and the "bulk" tensors in the middle A_2,A_3 are $D{\times}B{\times}B$-dimensional.

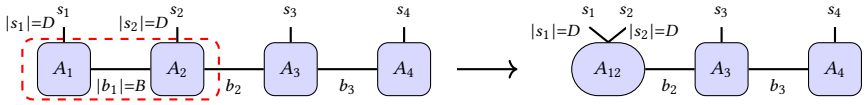


Fig. 3. Contraction of A_1 and A_2 nodes in the MPS results in a single tensor A_{12} with dimensions $D{\times}D{\times}B$.

the full $\mathbb{C}^{D^N}{\times}\mathbb{C}^{D^N}$ matrix would be infeasible for large N. In quantum mechanics, this is known as the "curse of dimensionality". To solve this issue, domain specialists decompose the Hamiltonian and the state vector into more compact TN representations of the problem and use eigensolvers that operate on a small group of local tensors of the TN.

3.1 Primer on TNs

3.1.1 Basic Concepts of TNs. TNs provide a mathematical framework for representing and manipulating extremely large tensors by factorizing them into networks of smaller tensors. In this network, nodes map to tensors, and the edges map to tensor dimensions. This framework mitigates the "curse of dimensionality," allowing many computations to be carried out with storage and time that scale polynomially with system size.

Figure 2 presents a factorization of the state vector $\vec{s}$ as a TN, where we label the node tensors as well as edge tensor dimensions and their sizes. The edges connected to a node determine the dimensions of the tensor. For example, node A_1 has a dimension of size D with label s_1, and a dimension of size B with label b_1. While the state vector $\vec{s}$ is of size D^N and grows exponentially with increasing N, this TN has $(N-2){\cdot}D{\cdot}B^2{+}2{\cdot}D{\cdot}B$ elements and grows polynomially.

In this TN representation, when an edge connects two nodes, it describes a possible multiply-accumulate operation over the dimension. Thus two TN nodes may be *contracted* or merged into one node, eliminating a node and an edge from a graph. For example, if A_1 and A_2 were contracted, the new node $A_{1,2}$ (shown graphically in Figure 3) stores the following $D{\times}D{\times}B$ tensor:

$$A_{1,2}[s_1,s_2,b_2] = \sum_{b_1=0}^{B} A_1[s_1,b_1]{\times}A_2[s_2,b_1,b_2]$$

This new node has three edges: two edges with size D and one edge with size B. If the TN is fully contracted to a single node, the resulting vector will be the $\vec{s} \in \mathbb{C}^{D^N}$.

3.1.2 TN Abstractions. Various TN algorithms operate on structured TNs that can be organized as a network of matrix product state (MPS) and matrix product operator (MPO) constructs.

The Bond Dimension. For both MPS and MPO networks, the size of the dimension shared between different tensors is a key parameter called the *bond dimension*. This parameter controls the allowed

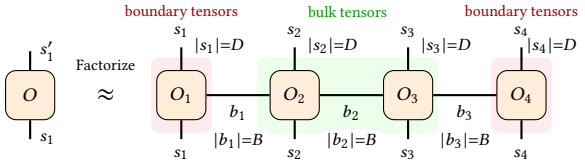


Fig. 4. MPO_O: Low-rank TN (MPO) representation of an operator O that maps an input state $\vec{s}$ to an output state $\vec{s}'$ over $N=4$ quantum variables, each with D states. The full operator $O \in \mathbb{C}^{D^N \times D^N}$ is factorized into a product of local tensors $\{O_k\}_{k=1}^{N}$. Each tensor O_k has two free physical indices corresponding to the input and output states of variable x_k (denoted s_k and s'_k), and shares a bond index b_k of dimension B with tensor O_{k+1}. The bond dimension B determines the number of ranks retained during factorization.

amount of interaction between system variables and thus determines the degrees of freedom of each tensor and the accuracy of the TN approximation. Higher bond dimension size allows for more accurate simulations, but longer simulation times, while lower bond dimensions result in lower accuracy for faster simulations. End-users can tune this parameter to trade simulation fidelity against computational cost.

Matrix Product States (MPS). A matrix product state (MPS) TN (Figure 2) is a low-rank decomposition of a vector. It is a 1D chain of tensor nodes where each node shares one common dimension with its neighbors and one free dimension. The dimension shared between all the different nodes are denoted by b_i. The size of this shared dimension, shown as B in the figure, is the bond dimension. The free edge of the node tensor A_i is the physical dimension denoted by s_i. This dimension determines the state assigned to variable x_i, and its size is equal to the number of states x_i can take (D). In a QOP, the state vector $\vec{s}$ is approximated as an MPS as follows:

$$\vec{s} \approx A_1[s_1,b_1]\left(\prod_{k=2}^{N-1} A_k[s_k,b_{k-1},b_k]\right)A_N[s_N,b_{N-1}]$$

When this MPS is contracted into a single node, it computes an approximation of $\vec{s}$. In this MPS, the size of shared indexes b_i (the bond dimension) is set by the user for the TN algorithm. In this MPS, the boundary tensors A_1,A_N are $D\times B$-dimensional tensors and all other A_i are $D\times B\times B$-dimensional tensors.

Matrix Product Operators (MPO). A matrix product operator (MPO) TN (Figure 4) is a low-rank decomposition of a matrix. It is a 1D chain of tensors where the i-th tensor determines how the full operator modifies the state of a quantum variable x_i. Similar to an MPS, in an MPO each tensor contains one shared dimension with its up to two neighbours; the size of the shared dimensions is the bond dimension. The MPO tensor, however, has two free indexes, which map the input state represented by the index s_i to the output state marked s'_i. When an MPO is contracted down into one node, the product of all the input indexes s_i and the output indexes s'_i forms the input/output dimensions of the original matrix. We describe an MPO for an operator matrix O as the following:

$$O \approx MPO_O[s \rightarrow s',(1,N)] = O_1[s_1,b_1,s'_1]\left(\prod_{k=2}^{N-1} O_k[s_k,b_{k-1},b_k,s'_k]\right)O_N[s_N,b_{N-1},s'_N]$$

The low-rank approximation of the Hamiltonian H can be represented with MPO_H, where the i'th tensor is given by H_i. This MPO computes the $\mathbb{C}^{D^N}\times\mathbb{C}^{D^N}$ Hamiltonian matrix H when fully contracted. Both the input/output indexes of the tensors H_i (given by s_i/s'_i) have dimension D (the number of states of the quantum variable) and the bond dimension size is B. For MPO_H, the boundary tensors H_1 and H_N are $D\times B\times D$-dimensional tensors and all other H_i are $D\times B\times B\times D$-dimensional tensors.

Composing MPSs/MPOs and calculating energy. MPSs and MPOs can be combined together to perform linear algebra operations, as shown in Figure 5. Stacking an MPO on top of an MPS

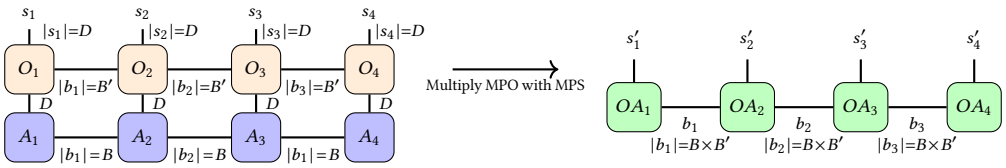


Fig. 5. Contraction of a MPS and MPO resulting in a new MPS with a higher bond dimension. The bond dimension can be reduced by using decomposition and truncation of low-weight singular values.

performs a matrix-vector multiplication over the decomposition of a matrix and a vector. Contracting nodes vertically executes the matrix-vector multiply. Stacking an MPO on top of an MPO performs a matrix-matrix multiplication over the decomposed matrices. Concatenating the Hamiltonian MPO to the state vector MPS over both sets of free indexes, allows us to compute the energy $\vec{s}^{\dagger} H \vec{s}$. Contracting this network fully results in a scalar value that captures the energy of the system.

3.2 Solving QOPs with TNs

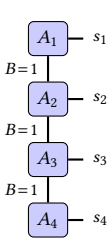
To mitigate the curse of dimensionality, TN-based eigensolvers operate directly on the TN representation of the system without fully expanding the network. These methods have controllable and exactly evaluable errors for large systems, where the degree of approximation is set through the bond dimension. In this formulation, a TN is constructed that computes $E(\vec{s}) = \vec{s}^{\dagger} H \vec{s}$, and various iterative TN algorithms exist to find the ground state by manipulating the TN, strategically contracting nodes to control memory usage in each iteration. These iterative algorithms produce an output MPS that approximates the eigenvector with the smallest eigenvalue for H. Two popular algorithms for such problems are the second generation Density Matrix Renormalization Group (DMRG) [62, 63] and Time-Evolving Block Decimation (TEBD)[61]. DMRG is widely used to find ground states of strongly coupled many-body systems and has also been extended to compute excited states and to simulate dynamical, finite-temperature, and non-equilibrium systems [25, 65].

3.3 Solving the Color Flipping QCOP

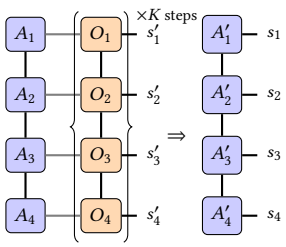
We will solve the color flipping example as an unconstrained QOP and a QCOP using the imaginary-time (it-)TEBD algorithm [56]. We use it-TEBD because it supports visualization of the intermediate state of the chain at each step. This algorithm modifies the state $\vec{s}$ by applying an MPO derived from the system Hamiltonian over sequential iterations. Each iteration suppresses high-energy components of the state relative to the ground state, resulting in convergence.

Encoding Color Flipping as a TN. We first encode the Color Flipping Game as an unconstrained TN over $N = 4$ quantum variables, each with $D = 2$ states corresponding to blue ($|0\rangle$) and red ($|1\rangle$). Rather than constructing the full $2^4 = 16$-dimensional state vector and 16×16 per-step operator applied by the solver, we approximate them as an MPS and an MPO, respectively. We initialize the MPS such that each site tensor A_i is set to $[1,1]$ (a uniform superposition of $|B\rangle$ and $|R\rangle$). This is an MPS with a bond dimension of 1 (Figure 6a). We choose this initial state to ensure that the starting state has at least some overlap with the solution state. The it-TEBD algorithm applies the MPO derived from the Hamiltonian tensor (MPO_O) over multiple time steps, as shown in Figure 6b, with a maximum allowed bond dimension of 16.

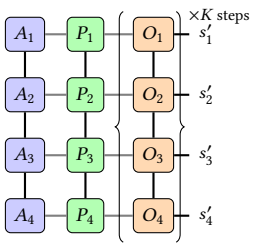
Figure 6d shows the probability of observing each state (y-axis) over the solver iterations (x-axis). Because the invalid states do not interact with the unconstrained Hamiltonian, their state coefficients remain unchanged initially. Over time, the valid states slowly start to dominate, pulling the system



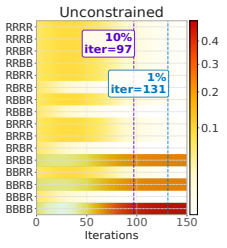
(a) Initial MPS



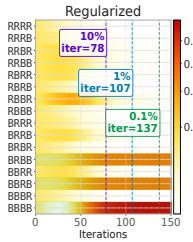
(b) it-TEBD: apply O for K steps



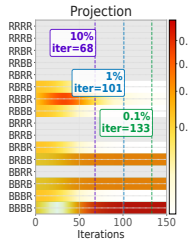
(c) With projection constraint



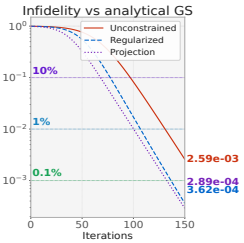
(d) Unconstrained



(e) Regularization



(f) Using Projection



(g) Infidelity

```
qvars x(N=4, D=2)
const Tensor d(2,2) = (1,0,0,0)
const Tensor X(2,2) = (0,1,1,0)
const Tensor T(8,8) =  d*X*d
for(i=1;i<N-2;i+=1):
  hamil_add(T@x(i)*x(i+1)*x(i+2))
for(i=1;i<N-1;i+=1):
  constr_add((x(i),x(i+1))!=(1,1))
```

(h) Masq code for Color Flipping Game with constraints

Fig. 6. Solving the Quantum Color Flipping Game with imaginary time (it-)TEBD. (a) Initial MPS with bond dimension 1, encoding all states as equally likely. (b) it-TEBD applies an MPO O derived from the Hamiltonian (orange) for $K=150$ steps to find the ground state MPS. (c) With projection: the projection MPO (green) prunes invalid states before it-TEBD. (d–f) State probabilities across iterations for the three different approaches, all converging to the ground state (Figure 1d). (g) Infidelity with the analytical ground state. (h) Masq implementation.

toward convergence. This formulation converges to the ground state with a 10% relative error after 97 iterations and a 1% after 131 iterations.

State Constraints with Regularization. To exploit the problem structure, we augment the Hamiltonian with penalty terms that discourage adjacent red-red configurations. Specifically, for each consecutive pair (x_i,x_{i+1}), we add a penalty term $\lambda|1\rangle\langle 1|\otimes|1\rangle\langle 1|$ with $\lambda=0.2$. This penalty artificially raises the energy of states where variables x_i,x_{i+1} are assigned the red-red state combination, biasing the evolution away from this inactive subspace. Figure 6e shows the result of TEBD with the augmented Hamiltonian. We see that the penalty terms actively suppress the invalid states step-by-step, reducing their effect much faster than the unconstrained formulation achieving a 10% relative error after 78 iterations and 1% error from ground state after 107 iterations.

State Constraints with Projection MPOs. While penalty terms bias the evolution away from invalid states, they only apply soft constraints and thus result in nonzero probability for invalid configurations during optimization. Domain-specialists have previously devised a specialized projection MPO (MPO_P) that implements the projection operator P (from Figure 1e) directly. This MPO zeros out the state configurations where any sequential pair of quantum variables is in red-red i.e. (R,R) configuration [32]. This MPO has bond dimension $B=2$, where this bond acts as a two-state automaton carrying the color of the current node to its neighbor: $b_i=0$ indicates blue and $b_i=1$ indicates red. A node in state $|R\rangle$ is only permitted if its left neighbor was blue, i.e. $b_{i-1}=0$. Applying this projection MPO once to the initial MPS before it-TEBD, as shown in Figure 6c, zeroes out all invalid configurations from the start. As seen in Figure 6f, the full it-TEBD optimization now operates entirely within the valid subspace and converges to a 10% error with ground state in just 68 iterations and 1% error in 101 iterations.

Building the Projection MPO. We describe the tensors that form the projection MPO: the first tensor P_1, the bulk tensors P_2,P_3, and the last tensor P_4. Multiplying the tensors of the projection MPO results in the constraint projector (Figure 1f) that zeros out all configurations containing adjacent $|R\rangle|R\rangle$ pairs.

The first tensor $P_1[s_1,b_1,s_1']$ records the color of x_1 into the bond b_1:

$$P_1 = 1 \text{ at } (0,0,0) \text{ and } (1,1,1); \quad 0 \text{ otherwise.}$$

The bulk tensors $P_i[s_i,b_{i-1},b_i,s_i']$ for $i \in \{2,3\}$ allow the variable x_i to be in state $|B\rangle$ regardless of the incoming bond, but allow $|R\rangle$ only when the previous node was $|B\rangle$ ($b_{i-1} = 0$):

$$P_i = 1 \text{ at } (0,j,0,0)\ \forall j \in \{0,1\} \text{ and } (1,0,1,1); \quad 0 \text{ otherwise.}$$

The last tensor $P_4[s_4,b_3,s_4']$ applies the same rule but has no outgoing bond:

$$P_4 = 1 \text{ at } (0,j,0)\ \forall j \in \{0,1\} \text{ and } (1,0,1); \quad 0 \text{ otherwise.}$$

Comparing Performance Figure 6g compares the relative error of the three schemes with respect to the true ground state over 150 iterations. Regularization-based formulation improves simulation accuracy and thus iteration counts over the unconstrained baseline. However, while the unconstrained case had a runtime of 0.65 seconds, Regularization requires an augmented Hamiltonian. This increases the computational cost of each it-TEBD step, resulting in a higher total runtime of 1.31 seconds. The projection MPO formulation results in the lowest simulation error and lowest iteration counts. It also does not artificially inflate the Hamiltonian, and has a total runtime of just 0.69 seconds.

3.4 Constraint Programming for TNs with CoTenN and Masq

CoTenN can automatically construct the full TN, including the projection MPO, given the Masq program for the QCOP in Figure 6h. In this program, line 1 declares a **`qvar`** creating a 4-dimensional array of 2-state quantum variables. Lines 2 and 3 declare the operators used in the Hamiltonian. Line 4 constructs the Hamiltonian via **`hamil_add`**, which CoTenN automatically translates into the it-TEBD Hamiltonian MPO. Line 5 encodes the state constraint via **`constr_add`**, which CoTenN automatically translates into the required projection MPO, synthesizing tensors P_1 through P_4, from just one line in Masq. CoTenN thus automatically synthesizes a TN implementation with an efficient constraint encoding.

4 Masq Programming Language

Masq is a domain-specific language for writing QCOPs that are efficiently mappable to TNs with CoTenN. A valid Masq constraint problem has the following properties:

- The QCOP can be efficiently mapped to a TN, solvable with TN algorithms like DMRG and TEBD.
- The QCOP constraints can be efficiently mapped to projection MPOs, which directly constrain the state vector to only take on valid states.

Core Language. Table 1 presents the grammar for the core language. Masq supports defining N-dimensional vectors of quantum variables with D states and constant tensors with **`qvar`** and **`const tensor`** constructs. Hamiltonians comprised of products of constant tensors and quantum variables are constructed with the **`hamil_add`** construct, and constraints may be added to the problem with the **`constr_add`** construct. Masq supports declaring global and local constraints. While a quantum variable can take multiple states, these constraints limit the valid state combinations allowed for the quantum variables involved in the constraint. Global constraints operate on a sequential subset of quantum variables at a time. Four kinds of global constraints are currently supported: (1) Sum constraint, (2) Eigenvalue constraint, (3) Not Equals constraint, and (4) Exactly one equals constraint. Local constraints operate over quantum variables with adjacent quantum variables. The constraint operator indexes are checked statically. Local constraints restrict the allowed states for sequential

quantum variables to obey some less than, greater than, less than equal to, greater than equal to, equality or inequality conditions. An example of a local constraint is the "$x(i) < x(i+1)$" syntax that imposes a constraint on quantum variables x_i,x_{i+1}, uch that the state amplitude for all configurations x_i,$x_{i+1} = (j,k)$ where $j,k \in \{0...D-1\}$ is 0 if $j \geq k$.

Static Checks. CoTenN performs several static checks to ensure Masq programs are well-formed:

▸ *Shape Checking.* Constant tensors are declared with a fixed shape and instantiated with data on declaration. CoTenN ensures that the size of the data corresponds to the shape of the tensor declared. The inline tensor data is read to be written in a row-major format and is reshaped into the required shape.

Multiplying tensors in Masq results in outer (Kronecker) products. Thus for Hamiltonian terms and writing eigenvalue constraints, if tensors are multiplied together, CoTenN computes the result of the specified outer product and confirms that the shape of the resulting tensor will be compatible with the shapes of the quantum variables it is multiplied with. Therefore, the dimension sizes of the result of the outer product need to match the dimension size of the state vector over the quantum variables in the product.

▸ *Index Checking.* For Masq programs, CoTenN makes two types of index checks. First, it checks that no quantum variable, an array, tuple, or a tensor is accessed at an invalid location. Next, to ensure successful compilation of constraints expressed in the Masq program, CoTenN makes two more assertions. All the variables used in a single constraint statement must belong to the same quantum variable name. Next, it ensures that any constraint statement is defined over only consecutive quantum variables. This means that if x_i is involved in the constraint expression, x_{i+2} can only be involved if x_{i+1} is also in the constraint. This ensures that the specified constraint can be lowered into an MPO form.

Python EDSL. The Masq language is implemented as a Python embedded domain-specific language. Therefore, Masq programs can be programmatically constructed using loops and other control flow, simplifying the process of encoding the constraint program. In the example code, we introduce numeric for loops of the form `for i in range(INT,INT,INT):` as syntactic sugar for constructing the Hamiltonian and problem constraints. In practice, these loops are unrolled into Masq statements. We also introduce syntactic sugar for building tensors out of well-known constant matrices, such as Identity I, Pauli matrices (X,Y,Z) for encoding of $D=2$ and more generally the spin matrices Sx,Sy,Sz.

4.1 CoTenN Compilation Flow and Execution Backend

The Masq program is compiled into a TN by CoTenN in three steps. First CoTenN maps all quantum variables in Masq into an MPS. The MPS characteristics like the dimension size for a variable and the number of variables are chosen based on the `qvar` arrays. If multiple arrays are declared, CoTenN concatenates them into one MPS by appending the variables of each array sequentially. Second, all constant tensors are stored and used to build the Hamiltonian MPO or the constraint MPOs when referenced. Third, CoTenN constructs the Hamiltonian MPO by processing the `hamil_add` statements. If the Hamiltonian contains only special operators (e.g., Sx,Sy,Sz), then it is lowered directly into backend code.

For each constraint, CoTenN processes all terms individually and lowers the constraint to a separate projection MPO that zeros out state combinations that violate the constraint specified. Applying multiple constraints in Masq results in them being enforced in conjunction. This is because CoTenN takes the entire set of constraint projection MPOs and multiplies them to get the total effective projection operator P. This operator is enforced for a TN eigensolver by first contracting it with the initial MPS. For DMRG, it is also applied to the Hamiltonian by calculating the effective Hamiltonian as $P^{\dagger}HP$. We further discuss how each constraint equation is lowered into a projection operator in Section 5. Our compilation flow generates code for the ITensors library [16], and uses its DMRG eigensolver implementation for our evaluation in Section 6. We use DMRG as the

Table 1. Grammer for MASQ

MASQ Grammar		
program	:=	QCOP{Block}
Block	:=	Stmt | Block; Stmt
Stmt	:=	qvar QID (N= NUM , D= NUM) | const tensor TensorID tuple<INT> = array<Complex> | hamil_add(TensorProd @ qvarProd) | constr_add(CstrStmt)
CstrStmt	:=	EignCstr | SumCstr | NotEqCstr | ExactlyOneEqsCstr | QRelationCstr
EignCstr	:=	TensorProd @ qvarProd == Complex * qvarProd
ExactlyOneEqsCstr	:=	ExactlyOneEqs(tuple<QID> , INT)
NotEqCstr	:=	tuple<QID> != tuple<INT>
SumCstr	:=	qvarSum rel_op INT
QRelationCstr	:=	term rel_op term
qvarSum	:=	INT * term + qvarSum | INT * term
rel_op	:=	< | > | <= | >= | == | !=
term	:=	QID [IDX]
qvarProd	:=	term * qvarProd | term
tuple<T>	:=	(list<T>)
list<T>	:=	T | list<T>, T
TensorProd	:=	TensorID* TensorProd | TensorID
array<T>	:=	[list<T>]
Complex	:=	NUM? +j NUM | NUM
QID, TensorID	:=	ID
ID	:=	`[A-Za-z_][A-Za-z0-9_]*`
INT	:=	`-?\d+`
NUM	:=	`-? \d+(\.\d+)?([eE][-+]?+)?`
IDX	:=	`\d+`

target eigensolver, as it is the standard method for computing ground states of one-dimensional and quasi-two-dimensional systems, which are the focus of our benchmarks.

4.2 Implementation

ITensors Backend. CoTenN targets ITensors[16] as the backend TN library due to its high-performance tensor contraction routines and widespread use in quantum simulation. CoTenN generates Julia code that expresses the Hamiltonian as a sum of terms; ITensors and other TN libraries can lower such expressions into an MPO directly. While we focus on ITensors in this work, CoTenN's tensor generation algorithms are backend-agnostic and can be extended to other TN frameworks such as TenPy or cuTensorNet.

Eigensolver Configuration File. To solve QCOPs encoded in MASQ, CoTenN generates code that invokes a target TN eigensolver. Different target benchmarks and problem sizes generally require different solver algorithms and settings. For example, DMRG requires specifying the number of sweeps, maximum bond dimension, and convergence thresholds which determine the target solution quality. Rather than hardcoding these choices directly in MASQ or the CoTenN compiler, CoTenN accepts an Eigensolver Configuration file that specifies the solver algorithm used and the corresponding configuration settings. This enables decoupling the solver configuration from the QCOP specification, with the compiler generating the appropriate invocation of the specified algorithm (e.g., applying DMRG with target bond dimensions and sweeps via ITensors [16]).

5 CoTenN Constraints

In this section, we define the constraints supported by MASQ (with their rule for Table 1 in brackets). All constraints are compiled into specialized MPOs called *projection MPOs* by CoTenN. A projection MPO acts on a subsequence of variables $(x_i)_{i\in S}$, where $S=[n,m]$, that all operate on the same number of states and masks out configurations that violate the target constraint, leaving all other variables unchanged. We also provide a quick reference for the notation used to describe QOPs, QCOPs (from Section 2), MPS and MPOs (from Section 3.1.2) and for constraint formulations for CoTenN in Table 2.

Table 2. Notation quick-reference for QOP, QCOP, MPS, MPO and and constraints.

Symbol	Description
Quantum Variables & State (Section 2.1)	
x_i	i-th quantum variable with D orthogonal basis states $\{0,\ldots,D-1\}$.
$\vec{s} \in \mathbb{C}^{D^N}$	Full state vector over all N variables.
$s[\mathbf{j}]$	Amplitude for configuration $\mathbf{j} = (j_1,\ldots,j_N)$ of $(x_1,\ldots,x_N)$; $P(\mathbf{j}) = \lvert\vec{s}[\mathbf{j}]\rvert^2$.
$H \in \mathbb{C}^{D^N \times D^N}$	Hamiltonian matrix encoding variable interactions.
$E(\vec{s}) = \vec{s}^{\dagger} H \vec{s}$	Energy of state $\vec{s}$; minimized at ground state.
$\vec{s}_{\text{opt}}$	Ground-state eigenvector (smallest eigenvalue of H).
Matrix Product States (MPS) (Section 3.1.2)	
A_i	MPS tensor at site i representing variable x_i.
$s_i = D$	Physical index of A_i, determining state of variable x_i; with $\lvert\lvert s_i \rvert\rvert = D$.
b_i	Bond index between tensors A_i and A_{i+1}; with $\lvert\lvert b_i \rvert\rvert = B$.
$A_i[s_i, b_{i-1}, b_i]$	Entry of bulk MPS tensor ($i \in (1,N)$) with shape $D \times B \times B$.
$A_1[s_1, b_1]$, $A_N[s_N, b_{N-1}]$	Entry of boundary MPS tensors with shape $D \times B$.
$\prod_i A_i$	Full MPS contraction $\approx \vec{s}$.
Matrix Product Operators (MPO) (Section 3.1.2)	
MPO_O	MPO encoding operator O.
O_i	MPO tensor for operator O acting on variable x_i.
s_i, s_i'	Input / output physical indices of O_i; dimension D.
$O_i[s_i, b_{i-1}, b_i, s_i']$	Entry of bulk MPO tensor ($i \in (1,N)$) with shape $D \times B_O \times B_O \times D$.
$O_1[s_1, b_1, s_1']$, $O_N[s_N, b_{N-1}, s_N']$	Entry of boundary MPO tensors with shape $D \times B_O \times D$.
Sparse Tensor Specification (Section 5.1)	
$\mathcal{A} = (Q, \Sigma, \{\delta_i\}_{i=n}^{m}, q_0, F)$	Step-dependent DFA lowered to a projection MPO.
$\text{supp}(T)$	Support set of tensor T: the set of index tuples where $T \neq 0$.
$\mathbb{1}[\cdot]$	Indicator function; 1 when predicate holds, 0 otherwise.

5.1 Finite Automata-Based CoTenN Constraints

Many of the constraints supported by CoTenN can be expressed as a *step-dependent deterministic finite automaton* $\mathcal{A} = (Q, \Sigma, \{\delta_i\}_{i=n}^{m}, q_0, F)$ which can then be lowered to a projection MPO. For this automaton, Q is a finite set of states, Σ is a finite alphabet, $\delta_i : Q \times \Sigma \to Q$ is the transition function at step i, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting states. Given an input sequence $(v_n,\ldots,v_m) \in \Sigma^{m-n+1}$, we define $q_{n-1} = q_0$ and compute $q_i = \delta_i(q_{i-1}, v_i)$ for $i = n,\ldots,m$ and accept if $q_m \in F$. This automaton automatically rejects a state for which some required valid transition (i.e., $\delta_i(q,v)$) is not defined. We provide a method to translate these automata to projection MPOs in this section, and describe constraints using this automaton formulation. We provide proofs for MPO correctness and the semantic corectness that the automata faithfully implement the target constraints in the paper's supplementary material.

From Finite Automata to Projection MPOs The automata used to describe constraints map directly to a binary-sparse projection MPO over quantum variables x_i for $i \in S$. The alphabet of the automaton corresponds to the set of values the target quantum variable can take, $\Sigma = \{0,\ldots,D-1\}$. The states of the automaton (Q) correspond to the possible values of the bond dimension, where the size of this bond dimension is given by $|Q|$. The i-th tensor of the MPO encodes the automaton transition δ_i. Each of the MPOs that can be described as automata leave the quantum state unchanged on accepted configurations and project the rejected configurations for the automaton to zero. The MPO tensors are binary-sparse: every entry is in $\{0,1\}$, and the input/output dimensions of these projection MPO tensors are always $s_i = s_i'$ for every nonzero entry. Such a binary-sparse tensor T is 1 at all positions in its support set ($supp(T)$), which is the set of index tuples where $T \neq 0$. Thus $T[\mathbf{idx}] = \mathbb{1}[\mathbf{idx} \in \text{supp}(T)]$.

▸ *First tensor.* $P_n[s_n, b_n, s_n']$ encodes the initial transitions from q_0 with transition function δ_n:

$$\text{supp}(P_n) = \{(v, \delta_n(q_0, v), v) \mid v \in \Sigma, \delta_n(q_0, v) \text{ is defined}\}$$

▸ *Bulk tensors.* $P_i[s_i,b_{i-1},b_i,s_i']$ for $i \in (n,m)$ encodes state-to-state transitions:

$$\text{supp}(P_i) = \{(v,q,\delta_i(q,v),v) \mid v \in \Sigma, q \in Q, \delta_i(q,v) \text{ is defined}\}$$

▸ *Last tensor.* $P_m[s_m,b_{m-1},s_m']$ encodes transitions that reach an accepting state:

$$\text{supp}(P_m) = \{(v,q,v) \mid v \in \Sigma, q \in Q, \delta_m(q,v) \in F\}$$

The corresponding projection MPO propagates the automaton state left to right using bond indexes.

$$MPO_P[s \to s',S] = P_n[s_n,b_n,s_n'] \times \prod_{i=n+1}^{m-1} P_i[s_i,b_{i-1},b_i,s_i'] \times P_m[s_m,b_{m-1},s_m']$$

Contracting this MPO is equivalent to running the automaton where only configurations that reach an accepting state survive. We now define each constraint by specifying its automaton.

5.1.1 Linear Algebra Constraint (SumCstr). The linear algebra constraint asserts the weighted sum of a group of quantum variables is less than or equal to W, where W and w_i are non-negative integers:

$$\sum_{i \in S} w_i x_i \le W$$

This constraint generalizes the capacity constraint from the Knapsack problem [26].

▸ *Automaton.* The automaton tracks the remaining capacity after processing each variable.

- **States:** $Q = \{0,1,\ldots,W\}$, representing the remaining capacity.
- **Initial state:** $q_0 = W$.
- **Transitions:** $\delta_i : (q,v) \mapsto q - w_i \cdot v$, defined only when $q - w_i \cdot v \ge 0$
- **Accepting states:** $F = Q$ (all states accepted; a sum greater than W results in invalid transition).

▸ *MPO Implementation.* This automaton is translated to the MPO_{LA} which will have a bond dimension of $W+1$. This constraint can also be easily extended for more cases: for strict equality the accepting states $F = \{0\}$, for strict inequality ($< W$), we can subtract 1 from W. For $\sum a_i x_i \ge W$, a symmetric constraint can be constructed which tracks the accumulated sum and accepts at the last site if $\ge W$.

5.1.2 Not Equals Constraint (NotEqCstr). This constraint asserts that a sequence of quantum variables is not equal to a target configuration $[w_n,\ldots,w_m]$ where each $w_i \in [0,D-1]$:

$$(x_n,x_{n+1},\ldots,x_m) \neq [w_n,w_{n+1},\ldots,w_m]$$

This constraint is derived from the PXP Model [27, 32].

▸ *Automaton.* The automaton tracks whether a mismatch has been seen.

- **States:** $Q = \{0,1\}$, where 1 means "all matched so far" and 0 means "at least one mismatch."
- **Initial state:** $q_0 = 1$.
- **Transitions:** $\delta_i : (q,v) \mapsto q$ for $v = w_i$; $\quad (q,v) \mapsto 0$ for $v \neq w_i$
- **Accepting states:** $F = \{0\}$ (accept only if at least one mismatch occurred).

▸ *MPO Implementation.* This can be mapped to a MPO with bond dimension of 2. CoTenN can in some special cases combine multiple Not Equals Constraint MPOs together when they all involve just 2 variables.

5.1.3 Exactly One Equals Constraint (ExactlyOneEqs). This constraint asserts that exactly one variable in a sequence equals $w \in [0,D-1]$:

$$\exists_{j \in [n,m]} (x_j = w) \cap \forall_{k \in S \setminus \{j\}} (x_k \neq w)$$

The travelling salesman problem [2] over D=N cities applies this constraint for all $w \in [0,N)$.

▸ *Automaton.* The automaton counts how many times w has been seen (up to 1).

- **States:** $Q = \{0,1\}$, where 0 means "not yet seen w" and 1 means "seen w exactly once."
- **Initial State:** $q_0 = 0$.
- **Transitions:** $\delta_i : (q,v) \mapsto q$ for $v \in \Sigma \setminus \{w\}$; $\quad (0,w) \mapsto 1 \quad (\delta_i(1,w)$ is not defined)

- **Accepting states:** $F=\{1\}$ (accept only if exactly one match occurred).

▸ *MPO Implementation.* This can be implemented with a bond dimension of 2. In case of travelling salesman with N cities and N time steps N such projection MPOs are applied.

5.1.4 Inequality Constraint (QRelationCstr). This constraint enforces a relational condition between two consecutive variables, for $op \in \{\geq,\leq,<,>,==,\neq\}$:

$$x_i \; op \; x_{i+1}$$

▸ *Automaton.* The automaton records the value of x_i and checks it against x_{i+1}.

- **States:** $Q=\{0,\ldots,D-1\}$, storing the value of the first variable.
- **Initial state:** $q_0=0$ (This state is unused as the transition just records v).
- **Transitions:** $\delta_i(q,v)=v$ (record x_i). $\delta_{i+1}(q,v)=q$, defined only when $q \; op \; v$.
- **Accepting states:** $F=Q$ (all states accept; rejection happens via undefined transitions in δ_{i+1}).

▸ *MPO Implementation.* This is a 2-node MPO with bond dimension equal to number of states for x_i (D).

5.2 Symmetry-Based Eigenvalue Constraints (EignCstr)

Unlike the generalized constraints, the eigenvalue constraint does not correspond to a classical finite automaton because the tensor entries are $D\times D$ operators rather than binary values. This constraint asserts that the state is an eigenvector of a Kronecker product of local operators with eigenvalue c:

$$\bigotimes_{i=n}^{m} T_i \cdot \vec{s} = c\vec{s}$$

▸ *MPO Implementation.* It can be implemented as MPO where the bond index selects a specific operator block on each site. The operator $\bigotimes_{i=n}^{m} T_i$ acts on variables $x_i \in [n,m]$ and can be formulated as:

$$MPO_{EG}[s \to s', S] = EG_n[s_n, b_n, s'_n] \times \prod_{i=n+1}^{m-1} EG_i[s_i, b_{i-1}, b_i, s'_i] \times EG_m[s_m, b_{m-1}, s'_m]$$

For an operator $\bigotimes_{i=n}^{m} T_i$ with eigenvalues C where $c \in C$, we get a product of MPOs that each prune an unwanted eigenvalue:

$$MPO_{EG} = \prod_{c' \in C \setminus \{c\}} MPO_{EG,c'} \qquad \text{where} \qquad MPO_{EG,c'} = \left(\bigotimes_{i} T_i - c' \cdot I\right)$$

▸ *MPO Node Tensors.* The node tensors are *block-sparse*: for each bond index value, the (s_i, s'_i) block is either a $D\times D$ operator or zero. For simplicity we show MPO_{EG} for the case $C=\{c,c'\}$. The first tensor $EG_n[s_n, b_n, s'_n]$, the tensors $EG_i[s_i, b_{i-1}, b_i, s'_i]$ where $i \in (n,m)$ and the last tensor $EG_m[s_m, b_{m-1}, s'_m]$ are:

$$EG_n[\cdot, b_n, \cdot] = \begin{cases} -c'I & \text{if } b_n = 0 \\ T_n & \text{if } b_n = 1 \end{cases} \quad EG_i[\cdot, b_{i-1}, b_i, \cdot] = \begin{cases} -c'I & \text{if } b_{i-1} = b_i = 0 \\ T_i & \text{if } b_{i-1} = b_i = 1 \end{cases} \quad EG_m[\cdot, b_{m-1}, \cdot] = \begin{cases} -c'I & \text{if } b_{m-1} = 0 \\ T_m & \text{if } b_{m-1} = 1 \end{cases}$$

6 Results

In this section, we discuss the advantages of using CoTenN generated TNs to solve QCOPs. We evaluate on two physics benchmarks with constraints expressed with Masq. In Section 6.1, we first describe the two benchmarks that can be understood as QCOPs. We describe the experimental setup in Section 6.2. Finally, we describe the speedups achieved by CoTenN for QCOPs in Section 6.3 and the benefits of using CoTenN for problems that can be modeled without constraints in Section 6.4.

6.1 Benchmark Physics Models

We evaluate CoTenN on two representative models that capture distinct and commonly occurring constraint classes: implicit symmetry-based and local structured constraints. Because there is no

```
qvars x(N=66, D=3)
const Tensor Hi(81, 81) = exp(i*pi*Sy)*exp(i*pi*Sx)*exp(i*pi*Sx)*exp(i*pi*Sy)
for(i=1; i<N; i+=2): hamil_add(Sz*Sz @ x(i)*x(i+1))
for(i=1; i<N-2; i+=2): hamil_add(Sx*Sx @ x(i)*x(i+3))
for(i=1; i<N-2; i+=2): hamil_add(Sy*Sy @ x(i+1)*x(i+2))
for(i=1;i<N-1;i+=2):
    constr_add(Hi@x(i)*x(i+1)*x(i+2)*x(i+3) = -1@x(i)*x(i+1)*x(i+2)*x(i+3))
```

Fig. 7. MASQ Program for **`Kitaev(1,1)`** with 32 Plaquettes (N=66 quantum variables with D=3 states).

```
qvars x(N=100, D=3)
const Tensor P01(3, 3) = [1, 0, 0, 0, 0, 0, 0, 0, 0]
const Tensor P10(3, 3) = [0, 0, 0, 0, 1, 0, 0, 0, 0]
const Tensor P00(3, 3) = [0, 0, 0, 0, 0, 0, 0, 0, 1]
const Tensor RP(3, 3) = [1, 0, 0, 0, 0, 0, 0, 0, 1]
const Tensor LP(3, 3) = [1, 0, 0, 0, 1, 0, 0, 0, 0]
const Tensor Xp(3, 3) = [0, 0, 1, 0, 0, 0, 1, 0, 0]
const Tensor Px(3, 3) = [0, 1, 0, 1, 0, 0, 0, 0, 0]
hamil_add(Xp @ x(1))
for(i=1; i<N-1; i+=2): hamil_add(Px*LP @ x(i)*x(i+1))
for(i=1; i<N-1; i+=2): hamil_add(RP*Xp @ x(i)*x(i+1))
hamil_add(Px @ x(N))
for(i=0; i<N-1; i+=1): constr_add((x(i), x(i+1)) != (1, 2))
```

Fig. 8. MASQ program for the PXP model with 200 atoms with spin 1 (D=3).

standardized physics benchmark set with with explicitly formalized constraints, we work with domain specialists to reformulate their physical intuitions as hard constraints, expressible with MASQ.

6.1.1 Kitaev Ladder Model. The Kitaev Ladder [10] is a simplified version of the Kitaev model on honeycomb lattice [29]. Higher-spin Kitaev Ladder cannot be solved exactly, and it requires algorithms like DMRG to find the ground state. Figure 7 presents a MASQ program for the Kitaev model for a 32 plaquette, spin 1 system with a target eigenvalue. The number of plaquettes determine the number of quantum variables, and the spin sets the number of states (spin $\frac{1}{2} \Rightarrow$ D=2, spin 1 $\Rightarrow$ D=3). The constraint specifies the target eigenvalue, and the shorthand **`Kitaev(z,k)`** is used to instantiate model with a target eigenvalue of **`k`** and a spin of **`z`**. We implement three Kitaev ladder variants with MASQ for evaluation:

- *Regularization:* The eigenvalue constraint is encoded with a regularization term into the Hamiltonian. This implementation is what is used for state-of-the-art simulations of the model.
- *Projection MPO:* The target eigenvalue constraint is encoded with a projection MPO using **`cstr_add`** (pictured above). This is a new problem encoding developed with COTENN's compiler.
- *Unconstrained:* The constraint is removed, producing a solution with a +1 eigenvalue.

▸ *In Prior Work.* The Kitaev Lattice model enforces plaquette symmetry constraints via eigenvalue conditions, expressible as `EignCstr` in MASQ. This constraint arises from commutation relationships on the Hamiltonian [10]. In prior work, these constraints are enforced only through regularization.

6.1.2 PXP Model. The PXP model is used to approximate Rydberg atoms in the "nearest-neighbor blockade regime" [27, 32]. In this scenario, each Rydberg atom can be modeled as a two-level system where only certain state oscillations are allowed. This spin constraint prevents adjacent atoms from simultaneously being in excited states. So two neighboring quantum variables cannot be in state $|11\rangle$. Figure 8 presents the MASQ program for the PXP model formulation that uses 3-state

quantum variables to model 2 state atoms to better capture spin constraints[32]. In our formulation, we construct a PXP model with 100 variables that map to 200 atoms.

- *Regularization:* The spin constraints are implemented as regularization terms and added to the Hamiltonian. This implementation is what is used for state-of-the-art simulations of this PXP model.
- *Projection MPO:* The spin constraints are implemented as a projection MPO using **`cstr_add`** (see above figure). This is a new problem encoding developed with CoTenN's compiler.

▸ *In Prior Work.* The PXP model [32] enforces local exclusion constraints on adjacent spins, expressible as `NotEqCstr` in Masq. These constraints naturally arise from the Rydberg blockade regime present in the material. Prior work provides a hand-constructed MPO for $D=2$ and a tensor expression (but not an MPO) for $D=3$. These hand-crafted MPOs are equivalent to our CoTenN-generated projection MPOs.

6.2 Experimental Setup

Benchmarks. To assess the impact of using CoTenN-generated projection MPO on simulator performance, we compare the Kitaev and PXP models that use Masq's built-in constraint operators (marked as **`Kitaev(z, k)-Proj`**/**`PXP-Proj`**), against baseline models that regularize the Hamiltonian (marked as **`Kitaev(z,k)-Reg`**/**`PXP-Reg`**). A penalty coefficient of $\lambda = 100.0$ is used for regularization. This penalty coefficient was chosen as it ensured the result did not violate the given constraints across problem sizes.
▸ *Regularization Baseline.* Regularization is a widely used approach for incorporating constraints in quantum simulation [5–7, 12, 22]: it suppresses invalid configurations by adding penalty terms to the Hamiltonian, and is broadly applicable even when no efficient bespoke encoding of the constraint is known. It thus serves as a natural baseline, as it represents the default approach available to practitioners without access to automated constraint compilation.
Experiments. In this work, we formulate two different ways of encoding constraints: using regularization and using automatically synthesized projection MPOs. Both are implemented in Masq and compiled through CoTenN to a TN that is then supplied to the ITensors [16] Julia library and solved with its DMRG eigensolver, which finds the ground states of the system. This ensures the performance differences reflect the choice of constraint encoding rather than implementation-level variation.

We run all experiments on an Intel 12th Gen i7-12700H CPU with a 64-bit x86 instruction set, 10 physical cores, and a 24MB LLC. The truncation cutoff value for DMRG is set to 10^{-10}, and the termination cutoff value is set to 10^{-4}. The truncation cutoff determines the minimum possible eigenvalue retained during decomposition for DMRG, and the termination cutoff determines when to stop the DMRG algorithm. For MPO projection executions, the DMRG algorithm is iteratively executed until the energy improvements diminish ($E_{Proj}[l-1] - E_{Proj}[l] < 10^{-4}$). For regularization executions, the DMRG algorithm runs until the energy is close enough to the projection execution's lowest energy ($E_{Reg}[l] - E_{Proj} < 10^{-4}$). This ensures a favorable stopping condition for the regularization baseline, allowing its execution to terminate early as soon as it can find a ground state with an energy level near the projection-based method's result. We also set a maximum number of iterations to ensure termination.

The total DMRG runtime can depend on the random initial MPS, the number of internal optimization operations per sweep, and system-level effects such as multithreaded scheduling. Thus, we run both configurations with 5 different random seeds and for each regularization run, use the target energy from the corresponding projection run with the same seed for all benchmarks.
Metrics. For each evaluation setting, we report the average number of DMRG iterations (Num Iters), the geometric mean time for the first DMRG sweep (Setup Avg) and the geometric mean time of the entire DMRG process (Total Avg). Overheads associated with starting Julia and JITing the program are excluded from these performance measurements. We compute for each run the Simulation Speedup ($(t_{\text{reg}} - t_{\text{reg,setup}})/(t_{\text{proj}} - t_{\text{proj,setup}})$) and the Total Speedup ($t_{\text{reg}}/t_{\text{proj}}$) for each run, where $t_{\text{reg}}, t_{\text{reg,setup}}$ are the total time and first sweep time for the regularized run and $t_{\text{proj}}, t_{\text{proj,setup}}$

Table 3. Performance of Regularized vs Projection MPO for Spin-1/2 Model.

Model Config.		`Kitaev(`$\frac{1}{2}$`, -1)-Reg`			`Kitaev(`$\frac{1}{2}$`, -1)-Proj`			**Speedup**	
# Vars	Bond Dim	Num Iters	Setup Avg	Total Avg	Num Iters	Setup Avg	Total Avg	Simulation $\frac{t_{\text{reg}}-t_{\text{reg,setup}}}{t_{\text{proj}}-t_{\text{proj,setup}}}$	Total $\frac{t_{\text{reg}}}{t_{\text{proj}}}$
34	10	54.00	17.35	30.41	6.20	6.83	9.05	4.96–7.88 (5.84×)	2.82–3.77 (3.31×)
	20	113.20	16.11	46.64	4.00	7.36	10.14	8.23–15.13 (11.07×)	3.79–6.05 (4.60×)
	40	169.20	18.44	158.30	4.00	8.00	13.57	22.09–28.83 (25.17×)	10.01–13.44 (11.67×)
	80	187.40	18.82	478.50	4.00	6.74	18.93	24.36–49.91 (38.89×)	18.25–33.09 (24.60×)
66	10	134.40	16.73	40.68	5.20	6.50	10.08	3.31–14.88 (6.42×)	2.72–7.19 (4.04×)
	20	180.60	18.25	131.53	5.00	6.32	10.65	15.83–55.85 (26.05×)	7.47–22.05 (12.34×)
	40	250.00	18.72	478.24	4.20	5.94	13.80	45.95–76.60 (58.75×)	28.31–39.58 (34.65×)
	80	250.00	19.12	1521.95	4.20	5.94	31.37	40.21–66.48 (59.27×)	34.95–53.32 (48.51×)
130	10	139.20	16.29	58.41	7.00	8.05	16.46	2.65–8.89 (4.84×)	2.26–5.43 (3.56×)
	20	160.80	17.84	235.45	6.00	8.82	23.71	10.83–19.04 (14.73×)	8.00–12.09 (9.93×)
	40	250.00	16.66	919.16	5.00	9.24	39.72	25.83–33.82 (29.66×)	19.50–26.97 (23.14×)
	80	250.00	17.94	3280.02	4.60	7.83	86.50	30.87–46.93 (39.40×)	28.85–41.97 (35.97×)

Table 4. Performance of Regularized vs Projection MPO for Spin-1 Model

Model Config.		`Kitaev(`1`, -1)-Reg`			`Kitaev(1, -1)-Proj`			**Speedup**	
# Vars	Bond Dim	Num Iters	Setup Avg	Total Avg	Num Iters	Setup Avg	Total Avg	Simulation $\frac{t_{\text{reg}}-t_{\text{reg,setup}}}{t_{\text{proj}}-t_{\text{proj,setup}}}$	Total $\frac{t_{\text{reg}}}{t_{\text{proj}}}$
34	10	61.4	16.85	33.24	9.2	8.62	13.55	1.94–8.50 (3.33×)	2.14–2.79 (2.45×)
	20	50.8	16.95	50.18	10.0	7.76	22.16	2.01–2.63 (2.30×)	2.05–2.37 (2.26×)
	40	99.4	16.98	163.86	10.0	8.50	54.51	2.69–3.64 (3.19×)	2.59–3.38 (3.01×)
	80	208.4	16.92	1213.52	10.0	7.83	189.28	4.87–7.99 (6.60×)	4.76–7.80 (6.41×)
66	10	92.0	17.92	53.49	10.0	8.02	18.50	2.38–6.84 (3.32×)	2.21–4.80 (2.89×)
	20	81.4	16.96	110.47	10.0	8.39	35.53	2.87–4.01 (3.45×)	2.67–3.58 (3.11×)
	40	153.2	16.86	486.22	10.0	8.95	119.62	3.48–5.27 (4.24×)	3.39–4.99 (4.06×)
	80	177.8	15.77	2206.77	10.0	9.163	460.86	3.58-5.72 (4.84×)	3.54–5.65 (4.78×)
130	10	91.4	19.17	90.03	10.0	8.382	27.28	2.59–8.08 (3.75×)	2.51–6.20 (3.30×)
	20	97.8	20.12	270.27	10.0	9.85	62.39	4.34–5.24 (4.76×)	3.95–4.70 (4.33×)
	40	150.2	16.44	895.08	10.0	10.79	201.10	4.12–4.92 (4.62×)	4.00–4.73 (4.45×)
	80	213.2	15.40	4938.35	10.0	10.00	914.61	5.32–6.90 (6.01×)	5.26–6.84 (5.96×)

is the total time and the first sweep time for the corresponding projection MPO-based run. For both Simulation and Total Speedup, we report the minimum, maximum and the geometric mean across runs (min–max (geomean)). To measure fidelity, we report the geometric mean of the truncation error (Trunc. Error) and system error (Sys. Error). Truncation error is the maximum amount of error arising from factorization during DMRG solve, and estimates how well a MPS approximates the exact solution. System error estimates the degree to which a constraint is violated. To report fidelity improvement, we compute the ratio of errors $\text{TruncErr}_{\text{Reg}}/\text{TruncErr}_{\text{Proj}}$, $\text{SysErr}_{\text{Reg}}/\text{SysErr}_{\text{Proj}}$.

6.3 Performance Analysis

6.3.1 Kitaev ladder Model. We compare the results of `Kitaev(`$\frac{1}{2}$`,-1)-Proj` to `Kitaev(`$\frac{1}{2}$`,-1)-Reg` in Table 3 and `Kitaev(1,-1)-Proj` with `Kitaev(1,-1)-Reg` in Table 4 across multiple spins, bond dimensions, and plaquette sizes. To ensure iso-accuracy comparisons, for each seed, the projection MPO program is executed first for a maximum of 10 iterations, and then the regularized program is executed for a maximum of 250 iterations until it finds a comparable solution. We report the average number of iterations, the geometric means of the setup and total times, and the min, max, and geometric mean of speedups.

Table 5. Performance of PXP Model for Regularized and projection MPO formulations with average ground state iterations (Grd Itrs) and excited state search iteration counts (Ext Itrs).

	PXP-Reg				**PXP-Proj**				**Speedup**	
Bond Dim	Grd Itrs	Ext Itrs	Setup Avg	Total Avg	Grd Itrs	Ext Itrs	Setup Avg	Total Avg	Simulation $\frac{t_{\text{reg}}-t_{\text{reg,setup}}}{t_{\text{proj}}-t_{\text{proj,setup}}}$	Total $\frac{t_{\text{reg}}}{t_{\text{proj}}}$
20	100	100	12.33	91.53	9.6	10	18.15	14.67	6.14–6.69 (6.43×)	4.85–5.22 (5.04×)
40	100	90.6	11.64	340.04	7.8	10	5.92	44.68	6.34–10.40 (7.75×)	5.82–9.14 (7.04×)
80	100	96.0	11.69	1167.97	6.6	10	6.21	158.43	6.56–8.28 (7.65×)	6.40–8.03 (7.43×)
160	100	100	12.93	5968.22	6.0	10	6.34	698.64	7.38–9.38 (8.60×)	7.34–9.30 (8.54×)

For **Kitaev(**$\frac{1}{2}$**,-1)**, the geometric mean of the simulation speedups ranges between 4.84×–59.2× and of the total speedups ranges between 3.31×–48.51× for **Proj** execution compared to the **Reg**-based TN execution. Across all evaluation settings and runs the simulation speedup varies between 2.65×–76.60× and total speedup varies between 3.31×–48.51×. For **Kitaev(**1 **,-1)**, the geometric mean for the simulation time speedup ranges between 2.30×–6.60× and for the total speedups ranges between 2.45×–6.41× for **Proj** execution compared to the **Reg**-based TN execution. Across all evaluation settings and runs, simulation speedups varies between 2.01×–8.50× and geomean total speedups vary between 2.05×–7.80×. Generally our performance numbers increase with increasing bond dimension.

For higher D problems, performance trends can be less predictable. One reason is that DMRG implementations for TN libraries generally do not enable sparse tensors, while using projection MPO operation can yield large sparse matrices, resulting in higher traffic. Using compressed representations for sparse Hamiltonians can substantially reduce their memory overhead and improve performance. ITensors library [16] has a block-sparse backend, though currently it is restricted to quantum-number constraints. Library developers are actively rewriting and expanding this infrastructure, with plans to expose it for more general use. Once these capabilities are available, CoTenN can directly target them to achieve additional performance gains.

6.3.2 PXP Model. Table 5 presents the performance improvements with **PXP-Proj** compared to **PXP-Reg** across bond dimensions ranging from 20 to 160. For each configuration, each **PXP** run calls DMRG twice: once to compute the ground state, and a second time with a penalty term to obtain the first excited state in order to calculate the energy gap. We cap the maximum number of iterations at 10 for the projection model and 100 for the regularized baseline. The table reports the average number of iterations required to reach the ground state (Grd Itrs) and excited state (Ext Itrs), along with the geometric mean of runtimes. As the bond dimension increases, the performance advantage of the **Proj** approach becomes more pronounced; the geometric mean speedup (averaged over 5 runs) increases from 6.43× to 8.60× for simulation time, and from 5.04× to 8.54× for total runtime. Across all configurations, the **Proj** executions consistently converge to the ground state in fewer than 10 iterations and find the excited state in 10 iterations. In contrast, the **Reg** executions require the full 100 iterations to reach a comparable ground state and on average 90.6—100 iterations to obtain the first excited state.

6.4 Analysis of Effect of Projections on Search Space

We next investigate the utility of the projection operator in narrowing the search space to find the unconstrained ground state. We compare the **Kitaev(z,+1)-Proj** and **Kitaev(z,+1)-Uncstr** executions for a 64 plaquette (130 variable) system across bond dimensions 10–80. For finding this solution state, **Proj** execution will only search over valid states, while the **Uncstr** execution searches over all states.

Table 6 presents the results for spin **z=**$\frac{1}{2}$, and Table 7 presents the results for spin **z=**1 Kitaev model. In the spin **z=**$\frac{1}{2}$ case, the geometric mean of the simulation speedup ranges between 2.47–20.80× and the geometric mean of total speedup ranges between 2.66–8.28×. We also achieve a $2.08 \cdot 10^3$

Table 6. Effect of projection MPO on ground state quality for spin $\frac{1}{2}$ Kitaev model with 64 plaquettes ($N = 130$).

		Bond Dimension			
Kitaev($\frac{1}{2}$, +1)	**Metric**	**10**	**20**	**40**	**80**
Unconstrained (**Unstr**)	Num Iters	250	5	7	10
	Setup Avg	17.816	24.931	23.182	17.235
	Total Avg	82.717	40.565	52.666	145.239
	Trunc. Error	$7.596 \cdot 10^{-5}$	$1.672 \cdot 10^{-6}$	$2.844 \cdot 10^{-8}$	$2.177 \cdot 10^{-10}$
	Sys. Error	$1.708 \cdot 10^{-3}$	$7.210 \cdot 10^{-5}$	$1.073 \cdot 10^{-3}$	$2.177 \cdot 10^{-10}$
MPO Projection (**Proj**)	Num Iters	3	3	3	3
	Setup Avg	7.027	6.871	6.882	8.034
	Total Avg	9.985	12.085	18.884	36.026
	Trunc. Error	$3.655 \cdot 10^{-8}$	$9.345 \cdot 10^{-12}$	$1.144 \cdot 10^{-15}$	$1.144 \cdot 10^{-15}$
	Sys. Error	$7.664 \cdot 10^{-12}$	$1.048 \cdot 10^{-12}$	$1.972 \cdot 10^{-10}$	$2.103 \cdot 10^{-10}$
	Metric	**10**	**20**	**40**	**80**
Aggregated Improvements	Simul. Speedup	18.58–24.43 (20.80×)	2.49–4.66 (2.97×)	2.14–2.89 (2.47×)	2.31–3.08 (2.63×)
	Total Speedup	7.87–9.06 (8.28×)	2.53–5.08 (3.26×)	2.37–3.15 (2.78×)	2.39–3.02 (2.66×)
	Trunc. Error Red.	$2.08 \cdot 10^{3}\times$	$1.79 \cdot 10^{5}\times$	$2.49 \cdot 10^{7}\times$	$1.90 \cdot 10^{5}\times$
	Sys. Error Red.	$2.23 \cdot 10^{8}\times$	$6.88 \cdot 10^{5}\times$	$5.44 \cdot 10^{6}\times$	$2.55 \cdot 10^{6}\times$

Table 7. Effect of projection MPO on ground state quality for spin 1 Kitaev model with 64 plaquettes ($N = 130$).

		Bond Dimension			
Kitaev(1, +1)	**Metric**	**10**	**20**	**40**	**80**
Unconstrained (**Unstr**)	Num Iters	250	250	8.4	10
	Setup Avg	17.582	17.807	17.220	17.688
	Total Avg	85.352	425.436	59.989	182.776
	Trunc. Error	$3.145 \cdot 10^{-3}$	$5.816 \cdot 10^{-4}$	$5.612 \cdot 10^{-5}$	$1.708 \cdot 10^{-6}$
	Sys. Error	$7.617 \cdot 10^{-5}$	$7.427 \cdot 10^{-6}$	$3.000 \cdot 10^{-7}$	$9.610 \cdot 10^{-4}$
MPO Projection (**Proj**)	Num Iters	9.6	9.8	10	10
	Setup Avg	7.807	8.393	9.119	9.805
	Total Avg	26.675	56.530	203.987	680.930
	Trunc. Error	$3.577 \cdot 10^{-4}$	$2.825 \cdot 10^{-4}$	$7.258 \cdot 10^{-6}$	$1.801 \cdot 10^{-7}$
	Sys. Error	$2.082 \cdot 10^{-6}$	$2.267 \cdot 10^{-8}$	$5.719 \cdot 10^{-8}$	$7.710 \cdot 10^{-8}$
	Metric	**10**	**20**	**40**	**80**
Aggregated Improvements	Simul. Speedup	0.61–6.11 (3.25×)	7.25–9.82 (8.45×)	0.18–0.31 (0.22×)	0.22–0.22 (0.22×)
	Total Speedup	1.17–4.93 (3.20×)	6.52–8.59 (7.50×)	0.25–0.39 (0.29×)	0.23–0.23 (0.23×)
	Trunc. Error Red.	0.88×	2.06×	7.73×	7.77×
	Sys. Error Red.	36.57×	$3.276 \cdot 10^{2}\times$	$5.906 \cdot 10^{4}\times$	$6.12 \cdot 10^{4}\times$

to $2.49 \cdot 10^7$ reduction in the geometric mean of the truncation error, and a $6.88 \cdot 10^5 - 2.23 \cdot 10^8$ reduction in geometric mean of the system error. Generally speaking, using projection MPOs enabled the TN to find better ground state estimates while using a smaller bond dimension. For the **z**=1 case, the geometric mean of the simulation speedups is 3.25×, 8.45× and the geometric mean of the total speedups is 3.20×, 7.50×, respectively, for smaller bond dimension cases (10, 20). We see high speedups with smaller bond dimensions as the **Proj** execution can find much better approximations of the ground state, while **Uncstr** was stuck in local minima due to the small bond dimension. For

Table 8. Summary of bespoke constraints from literature and CoTenN constraints that cover them.

Bespoke Constraint	Library/Solver Support			CoTenN Constraint		Our Contributions	
name	MPO form	ITensors	Solver	name	section	new MPO	General
Quantum Number [36, 43]	✓	✓	Any	Linear Algebra	Sec 5.1.1	✗	✓
constraint on total spin of system, number of particles ($\sum x_i = a$).							
Knapsack [54]	✓	✗	Any	Linear Algebra	Sec 5.1.1	✗	✓
capacity constraint for knapsack.							
Sum-Modulus [9, 18, 23, 45]	✗	✗	TEBD	Not Supported			
sum modulo 2 is zero ($\sum_i x_i \%2 = 0$). Used in pit Mining [18] optimization [23], physics simulation [9, 45].							
Global Internal Symmetry [51, 52]	✗	✗	TEBD	Not Supported			
all tensors in network are symmetric about some operation.							
Traveling Salesman [2]	✓	✗	Any	Exactly One	Sec. 5.1.3	✗	✓
constraint for a valid tour.							
Spin Constraint (D=2) [27, 32]	✓	✗	Any	Not Equals	Sec 5.1.2	✓	✓
constraint that ensure quantum variables have opposite spins. From the PXP model.							
Scheduling Problem [41]	✗	✗	Any	Inequality	Sec. 5.1.4	✓	✓
inquality constraint for resource constraint scheduling problems.							
Plaquette Symmetry [10]	✗	✗	Any†	Eigenvalue	Sec. 5.2	✓	✓
symmetry constraint for plaquette.							

larger bond dimensions, the **`Uncstr`** execution does not get similarly stuck and is able to find similar quality ground states for this unconstrained problem, allowing it to exit early. But even in these cases, using **`Proj`** formulations to restrict the search space results in much better solution quality. We reduce truncation error by 0.88–7.77× and the system error by $36.57\times$–$6.12 \cdot 10^4\times$ compared to the **`Uncstr`** execution. Furthermore, these improvements generally increase with bond dimension, resulting in improved truncation errors and better adherence to constraints with **`Proj`**.

7 Discussion: Constraint Coverage

Masq supports a broad class of constraints arising from conservation laws and symmetries of quantum systems. These constraints partition the Hilbert space into disjoint sectors, restricting the system to evolve within a valid sector. Multiple constraints can be composed in Masq to model systems with interacting symmetries. Each constraint expressed in Masq is compiled by CoTenN into a corresponding projection-MPO, providing a unified representation of constraints across different models and simplifying the construction of TN implementations. Table 8 summarizes the bespoke TN constraints from literature, and the corresponding generalized CoTenN constraint, and indicates whether CoTenN proposes a new MPO formulation (*new MPO*), or generalizes an older, manually crafted bespoke constraint (*general*).

Supported Constraints. CoTenN's Linear Algebra constraint generalizes the Quantum Number [36, 43, 53] and the Knapsack capacity constraints [54]. These constraints can enforce conservation laws such as fixed total spin or particle count; or for knapsack it can bound a weighted sum of variables. Both of these are instances of the linear inequalities $\sum a_i x_i \leq W$. CoTenN's NotEquals constraint was inspired from the PXP model's spin constraints [27, 32]. Prior work for this model only constructed a full MPO implementation for the $D = 2$ case; CoTenN supports arbitrary D with the generalized MPO formulation. CoTenN's ExactlyOneEquals constraint generalizes the valid tour constraint implemented for Traveling Salesman[2]. The Inequality Constraint MPO formulation is novel to this work. Finally the Eigenvalue Constraint appears in [10] but creating an MPO for this constraint is novel to this work.

If efficient hand-coded implementations exist for specific constraints, the design of Masq and CoTenN allows them to be readily incorporated. They can be supported as either special cases of existing constraints during compilation with CoTenN or as new constraint types in Masq.

Current Limitations. Our implementation does not yet capture all possible constraints. First challenge is that many constraints in physics are described implicitly, requiring manual translation into MASQ programs, which makes ensuring coverage challenging. Second, we do not support constraint classes for which we could not create efficient projection MPO implementations. For these constraints, prior work has proposed implementations that break MPS/MPO formulations, making them incompatible with standard solvers such as DMRG. These include global internal symmetry constraints, which require every tensor in the network to be individually symmetric under some operation [51, 52], and sum-modulus constraints (e.g., $\sum_i x_i \bmod 2 = 0$), which appear in pit mining [18], combinatorial optimization [23], and physics simulations [9, 45]. Finally, choosing system encoding (e.g., the local Hilbert space dimension D) remains manual. This choice affects both expressibility and performance, as seen with the PXP benchmark where a $D = 3$ encoding is preferred over $D = 2$ for simulation efficiency.

8 Future Work

Benchmark Set for QCOP Problems Our exploration suggests that many physically relevant constraints can be expressed within the MASQ and COTENN framework. For example, kinetically constrained dipole moment models [8, 14, 46, 47, 59] and lattice gauge theory (LGT) models [21, 33, 33, 57] involve constraints that can be expressed as linear algebraic or eigenvalue conditions. With the QCOP abstraction and the MASQ DSL, we can express more Hamiltonian simulation problems and build a corresponding benchmark set for this domain. Developing this benchmark set and extending support to additional constraint classes are important directions for future work.

Verification of TN Compilers COTENN currently only performs basic sanity checks on tensor dimensions and Hamiltonian structure, opening opportunities for deeper verification. First, key parameters like bond dimension and truncation thresholds are manually specified and directly affect approximation. Developing methods to bound approximation error and quantify its impact on constraint satisfaction and solution quality is an important direction for future work. Second, projection MPOs encode symmetries of the system Hamiltonian. Automatically verifying that these operators are structurally consistent with and commute with the Hamiltonian would provide stronger correctness guarantees. Third, practitioners often explore different system encodings (e.g., $D = 2$ vs. $D = 3$ in the PXP model), requiring modifications to both the Hamiltonian and constraint definitions. Automatically verifying equivalence across such encodings would help ensure alternative formulations preserve intended semantics.

Improved TN Algorithms for COTENN. COTENN generated TNs exhibit structured sparsity that is often not exploited by existing libraries. In particular, projection MPOs generated by COTENN have predictable sparsity patterns that can be leveraged to design specialized sparse tensor algebra kernels that reduce computation and memory overhead. Additionally, existing TN algorithms rely on heuristic contraction order selection that attempt to minimize intermediate tensor sizes. These heuristics are not sparsity aware, and building smarter cost models to find more efficient contraction orders could enable more performance.

9 Related Work

TN Libraries. Today, computational physicists directly implement their physics problems using TN libraries, such as ITensors[16] and cuTensorNet. These libraries supports construction of TNs comprised of MPS/MPO operators and offer different TN algorithms like DMRG [62, 63], TEBD [55], METTS [64] etc. Each of these algorithms have different specialties. However these libraries provide either very limited and specialized or no support for constraints. ITensors provides specialized library support only for quantum number constraints and requires users to manually construct tensor networks for other QCOPs. The cuTensorNet framework offers no specialized constraint support. Block2 [67] also supports different forms of symmetry constraints, like SU2, SO4, etc symmetries

Table 9. Constraint problem features supported by different problem encodings with emerging hardware.

Problem Encodings	Quantum Variable States	Higher-Order Interactions	Constraint Support	Probabilistic Ground State	Automated Mapping
Ising	2	✗	✗	✗	✓
hardware: p-computers, quantum computers, quantum annealers, oscillator-based computers					
Higher-Order Ising	2	✓	✗	✗	✓
hardware: p-computers, oscillator-based computers					
Potts Model	>2	✗	✗	✗	✗
hardware: oscillator-based computers					
Hamiltonian	2	✗	✗	✓	✓
hardware: quantum annealers, quantum computers					
Encoded Ising [24]	2	✓	✓	✗	✗
hardware: p-computers					
Problem Encodings for Commodity Hardware					
ILP/SMT/SAT Problems	N	✓	✓	✗	✓
Hamiltonian+Constraints	N	✓	✓	✓	✗
CoTenN/Masq	N	✓	✓*	✓	✓

for quantum chemistry problems. CoTenN in contrast, automatically constructs TNs that use projection MPOs for a large class of constraints. Unlike previous methods, it also enables composition of constraints using the constraint programming language Masq.

Hamiltonian Simulation Languages. While languages for hamiltonian simulations like SimuQ [40], Paulihedral [31], and HamLib [48] have been previously proposed, they generally target current quantum hardware and therefore encode all operators in the Pauli matrix basis, implicitly assuming qubit systems ($D=2$). This assumption is appropriate for near-term quantum devices, but it is a poor fit for TN simulation, where the variable encoding D is a first-class parameter and can be specified as such in Masq. Many physically important systems are intrinsically higher-dimensional: spin-1 systems (e.g., the KL model) requiring $D=3$ (qutrit basis $\{|0\rangle,|1\rangle,|2\rangle\}$). We also demonstrate this capability in our evaluation, where most of our benchmarks use a Spin-1 encoding with $D=3$.

Other Encodings and Solvers for QOPs/QCOPs Table 9 summarizes features of different constraint problems and supporting hardwares platforms. Except for Hamiltonian-based formulations, all constraint problem types expect ground state to resolve to deterministic values, making them ill-suited for physics problems. Unconstrained Hamiltonian-based problem encodings are implementable on quantum computers and annealers [1, 4, 68], using variational quantum eigensolvers, a class of hybrid quantum/classical algorithms [19, 30, 38, 42]. However current NISQ-era computers can not scale to large problems or reliably find global minima. They support constraints only through regularization and are limited to QCOPs with 2-state quantum variables. Regularization pollutes the search space with invalid states and introduces steep energy landscape regions, as observed by [24]. QCOP typically expressed with TNs with >2 states and constraints but such implementations lack automated mapping tools. CoTenN target this problem class with a compiler that automatically maps to TNs.

10 Conclusion

Simulation of physics problems is one of the most important use cases of quantum computing. In this work, we introduced a quantum constrained optimization problem (QCOP), a general abstraction for describing minimization problems over quantum variables that are subject to hard constraints. We presented Masq, the first constraint programming language for QCOPs implementable with TNs, and CoTenN, a compiler that automatically maps QCOPs specified with Masq programs to TNs. This framework serves as an initial step toward easing the development of physics simulations using TNs and facilitating re-use of bespoke constraints from literature.

Data Availability Statement

The code/artifact for this work is made available on Zenodo [50].

Acknowledgments

We thank Aaron Szasz for his great insights and help in collecting and understanding relevant benchmarks for the paper. We would also like to thank Shengtao Jiang for help with benchmarks for our work. We would like to thank Mark Horowitz for his advice. Cheng Peng was supported by the U.S. Department of Energy (DOE), Office of Basic Energy Sciences, Division of Materials Sciences and Engineering. We would also like to thank Dev Iyer, Pu (Luke) Yi and Kunal Sheth for their help in editing the paper. Furthermore, this work was also supported in part by the Stanford Center for Portable Accelerated Learning Hardware (Portal) as well as the SystemX Alliance. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the aforementioned funding agencies.

References

[1] Google Quantum AI and Collaborators. 2025. Quantum error correction below the surface code threshold. *Nature* 638 (2025), 920–926. https://doi.org/10.1038/s41586-024-08449-y Published online 09 December 2024; issue date 27 February 2025.

[2] Alejandro Mata Ali, Iñigo Perez Delgado, and Aitor Moreno Fdez. de Leceta. 2024. Traveling Salesman Problem from a Tensor Networks Perspective. arXiv:2311.14344 [quant-ph] https://arxiv.org/abs/2311.14344

[3] T. I. Andersen, N. Astrakhantsev, A. H. Karamlou, et al. 2025. Thermalization and criticality on an analogue–digital quantum simulator. *Nature* 638 (2025), 79–85. https://doi.org/10.1038/s41586-024-08460-3

[4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Michael Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergey V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander N. Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandra, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yu Niu, Eric Ostby, Andrei Petukhov, Jonathan C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amos Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Po Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574 (October 2019), 505–510. https://doi.org/10.1038/s41586-019-1666-5

[5] Fakher F. Assaad and Igor F. Herbut. 2013. Pinning the Order: The Nature of Quantum Criticality in the Hubbard Model on Honeycomb Lattice. *Phys. Rev. X* 3 (Aug 2013), 031010. Issue 3. https://doi.org/10.1103/PhysRevX.3.031010

[6] Matan Ben-Dov and Jing Chen. 2025. Regularized second-order optimization of tensor-network Born machines. arXiv:2501.18691 [cs.LG] https://arxiv.org/abs/2501.18691

[7] Miquel Albertí Binimelis. 2024. Quantum Annealing and Tensor Networks: a Powerful Combination to Solve Optimization Problems. arXiv:2412.05595 [quant-ph] https://arxiv.org/abs/2412.05595

[8] N. Cancrini, F. Martinelli, C. Roberto, and C. Toninelli. 2008. Kinetically constrained spin models. *Probability Theory and Related Fields* 140, 3-4 (2008), 459–504. https://doi.org/10.1007/s00440-007-0072-3

[9] Xie Chen, Bei Zeng, Zheng-Cheng Gu, Isaac L. Chuang, and Xiao-Gang Wen. 2010. Tensor product representation of a topological ordered phase: Necessary symmetry conditions. *Phys. Rev. B* 82 (Oct 2010), 165119. Issue 16. https://doi.org/10.1103/PhysRevB.82.165119

[10] Yushao Chen, Yin-Chen He, and Aaron Szasz. 2023. Phase diagrams of spin- <mml:math xmlns:mml="http://www.w3.org/1998/Math/MathML"><mml:mi>S</mml:mi></mml:math> Kitaev ladders. *Physical Review B* 108, 4 (July 2023). https://doi.org/10.1103/physrevb.108.045124

[11] T. A. Cochran, B. Jobst, E. Rosenberg, et al. 2025. Visualizing dynamics of charges and strings in (2+1)D lattice gauge theories. *Nature* 642 (2025), 315–320. https://doi.org/10.1038/s41586-025-08999-9

[12] Anurag Dwivedi, Miguel Angel Lopez-Ruiz, and Srinivasan S. Iyengar. 2024. Resource Optimization for Quantum Dynamics with Tensor Networks: Quantum and Classical Algorithms. *The Journal of Physical Chemistry A* 128, 32 (2024), 6774–6797. https://doi.org/10.1021/acs.jpca.4c03407

[13] Vincent E Elfving, Benno W Broer, Mark Webber, Jacob Gavartin, Mathew D Halls, K Patrick Lorton, and A Bochevarov. 2020. How will quantum computers provide an industrially relevant computational advantage in quantum chemistry? *arXiv preprint arXiv:2009.12472* (2020).

[14] Johannes Feldmeier and Michael Knap. 2021. Critically Slow Operator Dynamics in Constrained Many-Body Systems. *Phys. Rev. Lett.* 127 (Dec 2021), 235301. Issue 23. https://doi.org/10.1103/PhysRevLett.127.235301

[15] Richard P. Feynman. 1982. Simulating physics with computers. *International Journal of Theoretical Physics* 21, 6 (June 1982), 467–488. https://doi.org/10.1007/BF02650179

[16] Matthew Fishman, Steven R. White, and Edwin Miles Stoudenmire. 2020. The ITensor Software Library for Tensor Network Calculations. *ArXiv* abs/2007.14822 (2020). https://api.semanticscholar.org/CorpusID:220845808

[17] Martin Ganahl, Jackson Beall, Markus Hauru, Adam G.M. Lewis, Tomasz Wojno, Jae Hyeon Yoo, Yijian Zou, and Guifre Vidal. 2023. Density Matrix Renormalization Group with Tensor Processing Units. *PRX Quantum* 4 (Feb 2023), 010317. Issue 1. https://doi.org/10.1103/PRXQuantum.4.010317

[18] L. M. Giannini. 1991. Optimum design of open pit mines. *Bulletin of the Australian Mathematical Society* 43 (1991), 349 – 350. https://api.semanticscholar.org/CorpusID:121781746

[19] Shusuke Gocho, Hiroshi Nakamura, Sho Kanno, Yuya Nakagawa, and Naoki Yamamoto. 2023. Excited state calculations using variational quantum eigensolver with spin-restricted ansätze and automatically-adjusted constraints. *npj Computational Materials* 9, 1 (January 2023), 13. https://doi.org/10.1038/s41524-023-00965-1

[20] Sofía González-García, Aaron Szasz, Alice Pagano, Dvir Kafri, Guifré Vidal, and Agustin Di Paolo. 2025. Multi-Target Density Matrix Renormalization Group X algorithm and its application to circuit quantum electrodynamics. arXiv:2506.24109 [quant-ph] https://arxiv.org/abs/2506.24109

[21] Gaurav Gyawali, Shashwat Kumar, Yuri D. Lensky, Eliott Rosenberg, Aaron Szasz, Tyler Cochran, Renyi Chen, Amir H. Karamlou, Kostyantyn Kechedzhi, Julia Berndtsson, Tom Westerhout, Abraham Asfaw, Dmitry Abanin, Rajeev Acharya, Laleh Aghababaie Beni, Trond I. Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Nikita Astrakhantsev, Juan Atalaya, Ryan Babbush, Brian Ballard, Joseph C. Bardin, Andreas Bengtsson, Alexander Bilmes, Gina Bortoli, Alexandre Bourassa, Jenna Bovaird, Leon Brill, Michael Broughton, David A. Browne, Brett Buchea, Bob B. Buckley, David A. Buell, Tim Burger, Brian Burkett, Nicholas Bushnell, Anthony Cabrera, Juan Campero, Hung-Shen Chang, Zijun Chen, Ben Chiaro, Jahan Claes, Agnetta Y. Cleland, Josh Cogan, Roberto Collins, Paul Conner, William Courtney, Alexander L. Crook, Sayan Das, Dripto M. Debroy, Laura DeLorenzo, Alexander Del Toro Barba, Sean Demura, Agustin DiPaolo, Paul Donohoe, Ilya Drozdov, Andrew Dunsworth, Clint Earle, Alec Eickbusch, Aviv Moshe Elbag, Mahmoud Elzouka, Catherine Erickson, Lara Faoro, Reza Fatemi, Vinicius S. Ferreira, Leslie Flores Burgos, Ebrahim Forati, Austin G. Fowler, Brooks Foxen, Suhas Ganjam, Robert Gasca, William Giang, Craig Gidney, Dar Gilboa, Raja Gosula, Alejandro Grajales Dau, Dietrich Graumann, Alex Greene, Jonathan A. Gross, Steve Habegger, Michael C. Hamilton, Monica Hansen, Matthew P. Harrigan, Sean D. Harrington, Stephen Heslin, Paula Heu, Gordon Hill, Jeremy Hilton, Markus R. Hoffmann, Hsin-Yuan Huang, Ashley Huff, William J. Huggins, Lev B. Ioffe, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Cody Jones, Stephen Jordan, Chaitali Joshi, Pavol Juhas, Dvir Kafri, Hui Kang, Trupti Khaire, Tanuj Khattar, Mostafa Khezri, Mária Kieferová, Seon Kim, Paul V. Klimov, Andrey R. Klots, Bryce Kobrin, Alexander N. Korotkov, Fedor Kostritsa, John Mark Kreikebaum, Vladislav D. Kurilovich, David Landhuis, Tiano Lange-Dei, Brandon W. Langley, Pavel Laptev, Kim-Ming Lau, Loick LeGuevel, Justin Ledford, Joonho Lee, Kenny Lee, Brian J. Lester, Wing Yan Li, Alexander T. Lill, Wayne Liu, William P. Livingston, Aditya Locharla, Daniel Lundahl, Aaron Lunt, Sid Madhuk, Ashley Maloney, Salvatore Mandrà, Leigh S. Martin, Steven Martin, Orion Martin, Cameron Maxfield, Jarrod R. McClean, Matt McEwen, Seneca Meeks, Anthony Megrant, Xiao Mi, Kevin C. Miao, Amanda Mieszala, Sebastian Molina, Shirin Montazeri, Alexis Morvan, Ramis Movassagh, Charles Neill, Ani Nersisyan, Michael Newman, Anthony Nguyen, Murray Nguyen, Chia-Hung Ni, Murphy Yuezhen Niu, William D. Oliver, Kristoffer Ottosson, Alex Pizzuto, Rebecca Potter, Orion Pritchard, Leonid P. Pryadko, Chris Quintana, Matthew J. Reagor, David M. Rhodes, Gabrielle Roberts, Charles Rocque, Nicholas C. Rubin, Negar Saei, Kannan Sankaragomathi, Kevin J. Satzinger, Henry F. Schurkus, Christopher Schuster, Michael J. Shearn, Aaron Shorter, Noah Shutty, Vladimir Shvarts, Volodymyr Sivak, Jindra Skruzny, Spencer Small, W. Clarke Smith, Sofia Springer, George Sterling, Jordan Suchard, Marco Szalay, Alex Sztein, Douglas Thor, M. Mert Torunbalci, Abeer Vaishnav, Sergey Vdovichev, Guifre Vidal, Catherine Vollgraff Heidweiller, Steven Waltman, Shannon X. Wang, Theodore White, Kristi Wong, Bryan W. K. Woo, Cheng Xing, Z. Jamie Yao, Ping Yeh, Bicheng Ying, Juhwan Yoo, Noureldin Yosri, Grayson Young, Adam Zalcman, Yaxing Zhang, Ningfeng Zhu, Nicholas Zobrist, Sergio Boixo, Julian Kelly, Erik Lucero, Yu Chen, Vadim Smelyanskiy, Hartmut Neven, Dmitry Kovrizhin, Johannes Knolle, Jad C. Halimeh, Igor Aleiner, Roderich Moessner, and Pedram Roushan. 2025. Observation of disorder-free localization using a (2+1)D lattice gauge theory on a quantum processor. arXiv:2410.06557 [quant-ph] https://arxiv.org/abs/2410.06557

[22] Jingchang Hao, Zheng Zhu, and Yang Qi. 2026. Multi-target density matrix renormalization group for 3D CFTs on the fuzzy sphere. https://api.semanticscholar.org/CorpusID:285050987

[23] Tianyi Hao, Xuxin Huang, Chunjing Jia, and Cheng Peng. 2022. A Quantum-Inspired Tensor Network Algorithm for Constrained Combinatorial Optimization Problems. *Frontiers in Physics* 10 (2022). https://doi.org/10.3389/fphy.2022.906590

[24] Devrath Iyer and Sara Achour. 2025. Efficient Optimization with Encoded Ising Models. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 85–98. https://doi.org/10.1109/HPCA61900.2025.00018
[25] Eric Jeckelmann. 2002. Dynamical density-matrix renormalization-group method. *Phys. Rev. B* 66 (Jul 2002), 045114. Issue 4. https://doi.org/10.1103/PhysRevB.66.045114
[26] Richard Karp. 1972. Reducibility Among Combinatorial Problems. *Complexity of Computer Computations* 40, 85–103. https://doi.org/10.1007/978-3-540-68279-0_8
[27] Aron Kerschbaumer, Marko Ljubotina, Maksym Serbyn, and Jean-Yves Desaules. 2025. Quantum Many-Body Scars beyond the PXP Model in Rydberg Simulators. *Phys. Rev. Lett.* 134 (Apr 2025), 160401. Issue 16. https://doi.org/10.1103/PhysRevLett.134.160401
[28] Isaac H. Kim, Ye-Hua Liu, Sam Pallister, William Pol, Sam Roberts, and Eunseok Lee. 2022. Fault-tolerant resource estimate for quantum chemical simulations: Case study on Li-ion battery electrolyte molecules. *Phys. Rev. Res.* 4 (Apr 2022), 023019. Issue 2. https://doi.org/10.1103/PhysRevResearch.4.023019
[29] Alexei Kitaev. 2006. Anyons in an exactly solved model and beyond. *Annals of Physics* 321, 1 (2006), 2–111. https://doi.org/10.1016/j.aop.2005.10.005 January Special Issue.
[30] Thinh Viet Le and Vassilis Kekatos. 2024. Solving constrained optimization problems via the variational quantum eigensolver with constraints. *Phys. Rev. A* 110 (Aug 2024), 022430. Issue 2. https://doi.org/10.1103/PhysRevA.110.022430
[31] Gushu Li, Anbang Wu, Yunong Shi, Ali Javadi-Abhari, Yufei Ding, and Yuan Xie. 2022. Paulihedral: a generalized block-wise compiler optimization framework for Quantum simulation kernels. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS '22)*. Association for Computing Machinery, New York, NY, USA, 554–569. https://doi.org/10.1145/3503222.3507715
[32] Marko Ljubotina, Jean-Yves Desaules, Maksym Serbyn, and Zlatko Papić. 2023. Superdiffusive Energy Transport in Kinetically Constrained Models. *Phys. Rev. X* 13 (Mar 2023), 011033. Issue 1. https://doi.org/10.1103/PhysRevX.13.011033
[33] G. Magnifico, G. Cataldi, M. Rigobello, E. Calore, N. Schuch, S. Montangero, M. Dalmonte, E. Ercolessi, and G. Simone. 2025. Tensor networks for lattice gauge theories beyond one dimension. *Communications Physics* 8, 1 (2025), 322. https://doi.org/10.1038/s42005-025-02125-x
[34] G. Magnifico, T. Felser, P. Silvi, and et al. 2021. Lattice quantum electrodynamics in (3+1)-dimensions at finite density with tensor networks. *Nature Communications* 12 (2021), 3600. https://doi.org/10.1038/s41467-021-23646-3
[35] Salvatore Mandrà, Nikita Astrakhantsev, Sergei Isakov, Benjamin Villalonga, Brayden Ware, Tom Westerhout, and Kostyantyn Kechedzhi. 2025. A Heuristic for Matrix Product State Simulation of Out-of-Equilibrium Dynamics of Two-Dimensional Transverse-Field Ising Models. (11 2025). arXiv:2511.23438 [quant-ph]
[36] I. P. McCulloch. 2008. Infinite size density matrix renormalization group, revisited. (4 2008). arXiv:0804.2509 [cond-mat.str-el]
[37] T. L. Nguyen, J. M. Raimond, C. Sayrin, R. Cortiñas, T. Cantat-Moltrecht, F. Assemat, I. Dotsenko, S. Gleyzes, S. Haroche, G. Roux, Th. Jolicoeur, and M. Brune. 2018. Towards Quantum Simulation with Circular Rydberg Atoms. *Phys. Rev. X* 8 (Feb 2018), 011032. Issue 1. https://doi.org/10.1103/PhysRevX.8.011032
[38] Patrick J. J. O'Malley, Ryan Babbush, Ian D. Kivlichan, Jonathan Romero, Jarrod R. McClean, Rami Barends, Julian Kelly, Pedram Roushan, Andrew Tranter, Nan Ding, Brooks Campbell, Yu Chen, Z. Chen, B. Chiaro, A. Dunsworth, Austin Fowler, Evan Jeffrey, Erik Lucero, Anthony Megrant, Josh Mutus, Charles Neill, Chris Quintana, Daniel Sank, Amos Vainsencher, John Wenner, Theodore C. White, Peter V. Coveney, Peter J. Love, Hartmut Neven, Alán Aspuru-Guzik, and John M. Martinis. 2016. Scalable Quantum Simulation of Molecular Energies. *Physical Review X* 6, 3 (2016), 031007. https://doi.org/10.1103/PhysRevX.6.031007
[39] Tuure Orell, Alexios A. Michailidis, Maksym Serbyn, and Matti Silveri. 2019. Probing the many-body localization phase transition with superconducting circuits. *Phys. Rev. B* 100 (Oct 2019), 134504. Issue 13. https://doi.org/10.1103/PhysRevB.100.134504
[40] Yuxiang Peng, Jacob Young, Pengyu Liu, and Xiaodi Wu. 2024. SimuQ: A Framework for Programming Quantum Hamiltonian Simulation with Analog Compilation. *Proc. ACM Program. Lang.* 8, POPL, Article 81 (Jan. 2024), 31 pages. https://doi.org/10.1145/3632923
[41] Luis F. Pérez Armas, Stijn Creemers, and Sébastien Deleplanque. 2024. Solving the resource constrained project scheduling problem with quantum annealing. *Scientific Reports* 14 (2024), 16784. https://doi.org/10.1038/s41598-024-67168-6
[42] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. 2014. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications* 5 (July 2014), 4213. https://doi.org/10.1038/ncomms5213
[43] T. Pichler, M. Dalmonte, E. Rico, P. Zoller, and S. Montangero. 2016. Real-Time Dynamics in U(1) Lattice Gauge Theories with Tensor Networks. *Phys. Rev. X* 6 (Mar 2016), 011023. Issue 1. https://doi.org/10.1103/PhysRevX.6.011023
[44] Gokul Subramanian Ravi, Pranav Gokhale, Yi Ding, William Kirby, Kaitlin Smith, Jonathan M Baker, Peter J Love, Henry Hoffmann, Kenneth R Brown, and Frederic T Chong. 2022. CAFQA: A classical simulation bootstrap for variational quantum algorithms. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming*

Languages and Operating Systems, Volume 1. 15–29.
[45] N. Read and Subir Sachdev. 1991. Large-N expansion for frustrated quantum antiferromagnets. *Phys. Rev. Lett.* 66 (Apr 1991), 1773–1776. Issue 13. https://doi.org/10.1103/PhysRevLett.66.1773
[46] Felix Ritort and Peter Sollich. 2002. Glassy dynamics of kinetically constrained models. *Advances in Physics* 52 (2002), 219 – 342. https://api.semanticscholar.org/CorpusID:119486540
[47] Karl Royen, Suman Mondal, Frank Pollmann, and Fabian Heidrich-Meisner. 2024. Enhanced many-body localization in a kinetically constrained model. *Phys. Rev. E* 109 (Feb 2024), 024136. Issue 2. https://doi.org/10.1103/PhysRevE.109.024136
[48] Nicolas PD Sawaya, Daniel Marti-Dafcik, Yang Ho, Daniel P Tabor, David E Bernal Neira, Alicia B Magann, Shavindra Premaratne, Pradeep Dubey, Anne Matsuura, Nathan Bishop, Wibe A de Jong, Simon Benjamin, Ojas Parekh, Norm Tubman, Katherine Klymko, and Daan Camps. 2024. HamLib: A library of Hamiltonians for benchmarking quantum algorithms and hardware. *Quantum* 8 (Dec. 2024), 1559. https://doi.org/10.22331/q-2024-12-11-1559
[49] Philipp Scholl, Michael Schuler, Hannah J. Williams, Daniel Barredo, Sebastian Weber, Loïc Henriet, Vincent Lienhard, Immanuel Bloch, Stefan Kuhr, and Thierry Lahaye. 2021. Quantum simulation of 2D antiferromagnets with hundreds of Rydberg atoms. *Nature* 595 (2021), 233–238. https://doi.org/10.1038/s41586-021-03585-1
[50] Ritvik Sharma, Cheng Peng, Siddharth Dangwal, and Sara Achour. 2026. Masq and CoTenN. https://doi.org/10.5281/zenodo.19410701
[51] Sukhwinder Singh, Robert N. C. Pfeifer, and Guifré Vidal. 2010. Tensor network decompositions in the presence of a global symmetry. *Phys. Rev. A* 82 (Nov 2010), 050301. Issue 5. https://doi.org/10.1103/PhysRevA.82.050301
[52] Sukhwinder Singh and Guifre Vidal. 2012. Tensor network states and algorithms in the presence of a global SU(2) symmetry. *Phys. Rev. B* 86 (Nov 2012), 195114. Issue 19. https://doi.org/10.1103/PhysRevB.86.195114
[53] Sukhwinder Singh and Guifre Vidal. 2013. Global symmetries in tensor network states: Symmetric tensors versus minimal bond dimension. *Phys. Rev. B* 88 (Sep 2013), 115147. Issue 11. https://doi.org/10.1103/PhysRevB.88.115147
[54] Sergio Muñiz Subiñas, Jorge Martínez Martín, Alejandro Mata Ali, Javier Sedano, and Ángel Miguel García-Vico. 2025. Knapsack and Shortest Path Problems Generalizations From A Quantum-Inspired Tensor Network Perspective. arXiv:2506.11711 [quant-ph] https://arxiv.org/abs/2506.11711
[55] Masuo Suzuki. 1976. Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems. *Communications in Mathematical Physics* 51 (1976), 183–190. https://doi.org/10.1007/BF01609348
[56] Benjamin C. B. Symons, Dilhan Manawadu, David Galvin, and Stefano Mensa. 2024. Boosted imaginary time evolution of matrix product states. *Phys. Rev. B* 110 (Nov 2024), 174302. Issue 17. https://doi.org/10.1103/PhysRevB.110.174302
[57] L. Tagliacozzo, A. Celi, and M. Lewenstein. 2014. Tensor Networks for Lattice Gauge Theories with Continuous Groups. *Phys. Rev. X* 4 (Nov 2014), 041024. Issue 4. https://doi.org/10.1103/PhysRevX.4.041024
[58] Duc P. Truong, Mario I. Ortega, Ismael Boureima, Gianmarco Manzini, Kim Ø. Rasmussen, and Boian S. Alexandrov. 2024. Tensor networks for solving the time-independent Boltzmann neutron transport equation. *J. Comput. Phys.* 507 (2024), 112943. https://doi.org/10.1016/j.jcp.2024.112943
[59] Riccardo J. Valencia-Tortora, Nicola Pancotti, and Jamir Marino. 2022. Kinetically Constrained Quantum Dynamics in Superconducting Circuits. *PRX Quantum* 3 (Jun 2022), 020346. Issue 2. https://doi.org/10.1103/PRXQuantum.3.020346
[60] Steven Vancoillie, Per Åke Malmqvist, and Valera Veryazov. 2016. Potential energy surface of the chromium dimer re-re-revisited with multiconfigurational perturbation theory. *Journal of chemical theory and computation* 12, 4 (2016), 1647–1655.
[61] G. Vidal. 2007. Classical Simulation of Infinite-Size Quantum Lattice Systems in One Spatial Dimension. *Phys. Rev. Lett.* 98 (Feb 2007), 070201. Issue 7. https://doi.org/10.1103/PhysRevLett.98.070201
[62] Steven R. White. 1992. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.* 69 (Nov 1992), 2863–2866. Issue 19. https://doi.org/10.1103/PhysRevLett.69.2863
[63] Steven R. White. 1993. Density-matrix algorithms for quantum renormalization groups. *Phys. Rev. B* 48 (Oct 1993), 10345–10356. Issue 14. https://doi.org/10.1103/PhysRevB.48.10345
[64] Steven R. White. 2009. Minimally Entangled Typical Quantum States at Finite Temperature. *Phys. Rev. Lett.* 102 (May 2009), 190601. Issue 19. https://doi.org/10.1103/PhysRevLett.102.190601
[65] Steven R. White and Adrian E. Feiguin. 2004. Real-Time Evolution Using the Density Matrix Renormalization Group. *Phys. Rev. Lett.* 93 (Aug 2004), 076401. Issue 7. https://doi.org/10.1103/PhysRevLett.93.076401
[66] M. Xu, L. H. Kendrick, A. Kale, et al. 2025. A neutral-atom Hubbard quantum simulator in the cryogenic regime. *Nature* 642 (2025), 909–915. https://doi.org/10.1038/s41586-025-09112-w
[67] Huanchen Zhai, Henrik R. Larsson, Seunghoon Lee, Zhi-Hao Cui, Tianyu Zhu, Chong Sun, Linqing Peng, Ruojing Peng, Ke Liao, Johannes Tölle, Junjie Yang, Shuoxue Li, and Garnet Kin-Lic Chan. 2023. Block2: A comprehensive open source framework to develop and apply state-of-the-art DMRG algorithms in electronic structure and beyond. *The Journal of Chemical Physics* 159, 23 (2023), 234801. https://doi.org/10.1063/5.0180424

[68] C. Zhang, R. G. Cortiñas, A. H. Karamlou, N. Noll, J. Provazza, J. Bausch, S. Shirobokov, A. White, M. Claassen, S. H. Kang, A. W. Senior, N. Tomašev, J. Gross, K. Lee, T. Schuster, W. J. Huggins, H. Celik, A. Greene, B. Kozlovskii, F. J. H. Heras, A. Bengtsson, A. Grajales Dau, I. Drozdov, B. Ying, W. Livingstone, V. Sivak, N. Yosri, C. Quintana, D. Abanin, A. Abbas, R. Acharya, L. Aghababaie Beni, G. Aigeldinger, R. Alcaraz, S. Alcaraz, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, W. Askew, N. Astrakhantsev, J. Atalaya, B. Ballard, J. C. Bardin, H. Bates, M. Bigdeli Karimi, A. Bilmes, S. Bilodeau, F. Borjans, A. Bourassa, J. Bovaird, D. Bowers, L. Brill, P. Brooks, M. Broughton, D. A. Browne, B. Buchea, B. B. Buckley, T. Burger, B. Burkett, J. Busnaina, N. Bushnell, A. Cabrera, J. Campero, H. S. Chang, S. Chen, Z. Chen, B. Chiaro, L. Y. Chih, A. Y. Cleland, B. Cochrane, M. Cockrell, J. Cogan, R. Collins, P. Conner, H. Cook, W. Courtney, A. L. Crook, B. Curtin, S. Das, M. Damyanov, D. M. Debroy, L. De Lorenzo, S. Demura, L. B. De Rose, A. Di Paolo, P. Donohoe, A. Dunsworth, V. Ehimhen, A. Eickbusch, A. M. Elbag, L. Ella, M. Elzouka, D. Enriquez, C. Erickson, V. S. Ferreira, M. Flores, L. Flores Burgos, E. Forati, J. Ford, A. G. Fowler, B. Foxen, M. Fukami, A. W. L. Fung, L. Fuste, S. Ganjam, G. Garcia, C. Garrick, R. Gasca, H. Gehring, R. Geiger, É. Genois, W. Giang, C. Gidney, D. Gilboa, J. E. Goeders, E. C. Gonzales, R. Gosula, S. J. de Graaf, D. Graumann, J. Grebel, J. Guerrero, J. D. Guimarães, T. Ha, S. Habegger, T. Hadick, A. Hadjikhani, M. P. Harrigan, S. D. Harrington, J. Hartshorn, S. Heslin, P. Heu, O. Higgott, R. Hiltermann, J. Hilton, H. Y. Huang, M. Hucka, C. Hudspeth, A. Huff, E. Jeffrey, S. Jevons, Z. Jiang, X. Jin, C. Joshi, P. Juhas, A. Kabel, H. Kang, K. Kang, R. Kaufman, K. Kechedzhi, T. Khattar, M. Khezri, S. Kim, R. King, O. Kiss, P. V. Klimov, C. M. Knaut, B. Kobrin, F. Kostritsa, J. M. Kreikebaum, R. Kudo, B. Kueffler, A. Kumar, V. D. Kurilovich, V. Kutsko, N. Lacroix, D. Landhuis, T. Lange-Dei, B. W. Langley, P. Laptev, K. M. Lau, L. Le Guevel, J. Ledford, J. Lee, B. J. Lester, W. Leung, L. Li, W. Y. Li, M. Li, A. T. Lill, M. T. Lloyd, A. Locharla, D. Lundahl, A. Lunt, S. Madhuk, A. Maiti, A. Maloney, S. Mandra, L. S. Martin, O. Martin, E. Mascot, P. Masih Das, D. Maslov, M. Mathews, C. Maxfield, J. R. McClean, M. McEwen, S. Meeks, K. C. Miao, R. Molavi, S. Molina, S. Montazeri, C. Neill, M. Newman, A. Nguyen, M. Nguyen, C. H. Ni, M. Y. Niu, L. Oas, R. Orosco, K. Ottosson, A. Pagano, S. Peek, D. Peterson, A. Pizzuto, E. Portoles, R. Potter, O. Pritchard, M. Qian, A. Ranadive, M. J. Reagor, R. Resnick, D. M. Rhodes, D. Riley, G. Roberts, R. Rodriguez, E. Ropes, E. Rosenberg, E. Rosenfeld, D. Rosenstock, E. Rossi, D. A. Rower, M. S. Rudolph, R. Salazar, K. Sankaragomathi, M. C. Sarihan, K. J. Satzinger, M. Schaefer, S. Schroeder, H. F. Schurkus, A. Shahingohar, M. J. Shearn, A. Shorter, N. Shutty, V. Shvarts, S. Small, W. C. Smith, D. A. Sobel, R. D. Somma, B. Spells, S. Springer, G. Sterling, J. Suchard, A. Szasz, A. Sztein, M. Taylor, J. P. Thiruraman, D. Thor, D. Timucin, E. Tomita, A. Torres, M. M. Torunbalci, H. Tran, A. Vaishnav, J. Vargas, S. Vdovichev, G. Vidal, C. Vollgraff Heidweiller, M. Voorhees, S. Waltman, J. Waltz, S. X. Wang, B. Ware, J. D. Watson, Y. Wei, T. Weidel, T. White, K. Wong, B. W. K. Woo, C. J. Wood, M. Woodson, C. Xing, Z. J. Yao, P. Yeh, J. Yoo, E. Young, G. Young, A. Zalcman, R. Zhang, Y. Zhang, N. Zhu, N. Zobrist, Z. Zou, G. Bortoli, S. Boixo, J. Chen, Y. Chen, M. Devoret, M. Hansen, C. Jones, J. Kelly, P. Kohli, A. Korotkov, E. Lucero, J. Manyika, Y. Matias, A. Megrant, H. Neven, W. D. Oliver, G. Ramachandran, R. Babbush, V. Smelyanskiy, P. Roushan, D. Kafri, R. Sarpong, D. W. Berry, C. Ramanathan, X. Mi, C. Bengs, A. Ajoy, Z. K. Minev, N. C. Rubin, and T. E. O'Brien. 2025. Quantum computation of molecular geometry via many-body nuclear spin echoes. arXiv:2510.19550 [quant-ph] https://arxiv.org/abs/2510.19550

Received 2025-11-14; accepted 2026-04-03