Parity Quantum Optimization: Compiler

Kilian Ender[1,2], Roeland ter Hoeven[1,2], Benjamin E. Niehoff[1], Maike Drieb-Schön[1,2], and Wolfgang Lechner[1,2]

[1]Parity Quantum Computing GmbH, A-6020 Innsbruck, Austria

[2]Institute for Theoretical Physics, University of Innsbruck, A-6020 Innsbruck, Austria

We introduce parity quantum optimization with the aim of solving optimization problems consisting of arbitrary k-body interactions and side conditions using planar quantum chip architectures. The method introduces a decomposition of the problem graph with arbitrary k-body terms using generalized closed cycles of a hypergraph. Side conditions of the optimization problem in form of hard constraints can be included as open cycles containing the terms involved in the side conditions. The generalized parity mapping thus circumvents the need to translate optimization problems to a quadratic unconstrained binary optimization problem (QUBO) and allows for the direct encoding of higher-order constrained binary optimization problems (HCBO) on a square lattice and full parallelizability of gates.

1 Introduction

Hard optimization problems are omnipresent in all fields of science, technology, and industry [1, 2]. Classical algorithms for solving optimization problems have been studied and refined over decades and are now at a mature stage where fundamental improvements are unlikely. An answer to the pressing need for faster and more efficient optimization algorithms may be provided by quantum devices. The impressive recent progress in building highly coherent quantum computers based on ions [3–5], atoms [6–10], superconducting circuits [11–14], crystal defects [15], and photonic systems [16, 17] has led to a considerable interest in quantum algorithms to solve hard optimization problems. Quantum optimization algorithms, be they digital [18, 19] or analog [20–22], employ radically new paradigms of computation. Their basic principle is to encode an optimization problem as a physical quantum system whose energy corresponds to the problem's cost function. Solving the optimization problem is then equivalent to finding the lowest-energy configuration of the physical system. Recently, quantum algorithms for solving optimization problems have been developed for both analog and digital quantum computers [23–26]. Their possible speedup compared to classical algorithms, scaling and robustness to error, however, are open questions.

Wolfgang Lechner: wolfgang@parityqc.com, wolfgang.lechner@uibk.ac.at,
K. Ender, R. ter Hoeven and B.E. Niehoff contributed equally.

Here, we present a novel abstraction of optimization problems based on the parity transformation. This transformation maps a subset of k spins into a single parity qubit whose value is equal to their k-fold product. With this transformation, an optimization problem of arbitrary connectivity can be encoded in a physical system with only *local* connectivity. The parameters of the optimization problem become local magnetic fields, and the consistency conditions of the mapping become 3- or 4-body physical interactions whose strengths are independent of the logical problem. For a wide variety of problems, it is then possible to lay out the parity-transformed qubits in a 2-dimensional grid on a physical device such that all the necessary couplings can be implemented. The technique can just as well encode constrained optimization problems with various types of constraints. We present the mathematical foundations of the parity transformation, which allows one to encode optimization problems with arbitrary k-body interaction terms onto a realistic quantum computing device, along with examples of device layouts obtained by the compiler employing this transformation.

We consider optimization problems represented by a Hamiltonian of the form

$$H = \sum_{i=1}^{N} J_i \sigma_z^{(i)} + \sum_{i=1}^{N} \sum_{j>i} J_{ij} \sigma_z^{(i)} \sigma_z^{(j)} + \sum_{i=1}^{N} \sum_{j>i} \sum_{k>j} J_{ijk} \sigma_z^{(i)} \sigma_z^{(j)} \sigma_z^{(k)} + \dots, \tag{1}$$

which contains arbitrary higher-order k-body spin terms. In the current state-of-the-art methods, these higher-order k-body terms would have to be decomposed into quadratic terms using (a potentially large number of) auxiliary qubits. Our proposal instead allows k-body terms to be directly implemented in the physical layout, omitting the need for the intermediate mapping to quadratic unconstrained binary optimization (QUBO).

In addition to this Hamiltonian, we consider impos-

ing constraints of the form

$$C(\{\sigma_z^{(i)}\}) = \delta_0 + \sum_{i=1}^{N} c_i \sigma_z^{(i)} + \sum_{i=1}^{N} \sum_{j>i} c_{ij} \sigma_z^{(i)} \sigma_z^{(j)} \quad (2)$$
$$+ \sum_{i=1}^{N} \sum_{j>i} \sum_{k>j} c_{ijk} \sigma_z^{(i)} \sigma_z^{(j)} \sigma_z^{(k)} + \ldots,$$

where one now seeks to optimize over the constrained subspace of states satisfying $C(\{\sigma_z^{(i)}\})\,|\psi\rangle = 0$. These constraints amount to arbitrary polynomials of the spin variables $C(\{s_z^{(i)}\}) = 0$ for $s_z^{(i)} = \pm 1$ (provided that the coefficients δ_0, c_i, c_{ij}, $c_{ijk}, \ldots$ are consistent with the existence of solutions). This covers the cases of product constraints and sum constraints, as well as more general constraints [27].

In this paper we introduce the parity transformation including problems with product constraints. In the associated publication [27] we describe the implementation of general constraints and in Ref. [28] we benchmark the resource requirements for the quantum approximate optimization algorithm in the parity architecture for various classes of problems.

2 The Parity Transformation

The parity transformation is the generalization of the LHZ mapping [29] for hypergraphs and side conditions in the optimization problem. Let us review the LHZ mapping for completeness: the mapping is a representation of all-to-all graphs with k-body interactions using parity variables. For all-to-all pair interactions between N logical qubits, the LHZ representation is a 2D square lattice with 4-body interactions and $N(N-1)/2$ physical qubits. The 4-body interactions are derived from closed cycles in the logical graph. The condition of closed cycles can be understood from the following considerations: the physical qubits $\tilde{\sigma}_z^{(ij)} = \sigma_z^{(i)} \sigma_z^{(j)}$ represent the product of two logical qubits i and j. The product of four such physical qubits in a closed cycle is thus of the form $\sigma_z^{(i)} \sigma_z^{(j)} \sigma_z^{(j)} \sigma_z^{(k)} \sigma_z^{(k)} \sigma_z^{(l)} \sigma_z^{(l)} \sigma_z^{(i)} = 1$, which always comes out to $+1$ as all indices must appear twice.

In [29] a slightly generalized LHZ construction is also presented, which maps a 3-body all-to-all graph onto a 3D cubic lattice, with 4-body cyclic constraints arranged on its faces. In principle such a construction can be generalized to k-body all-to-all problems, putting them on a k-dimensional k-cubic lattice, although this is impractical for realistic devices. In general such constructions require a number of lattice qubits scaling as N^k; for 2D devices, this means one can represent at best 2-body all-to-all problems, using $O(N^2)$ physical qubits. We present here a further generalization, called the 'parity transformation', that can represent *arbitrary* optimization problems (that is, with mixtures of k-body terms of various k) on a 2D square lattice (that is, without resorting to higher-dimensional constructions). The required number of physical qubits scales as K, where K is the number of non-zero terms in the Hamiltonian. The same mapping can also encode constrained optimization problems without overhead. There is no need to rewrite the Hamiltonian using QUBO-like transformations, as the k-body interactions are encoded directly.

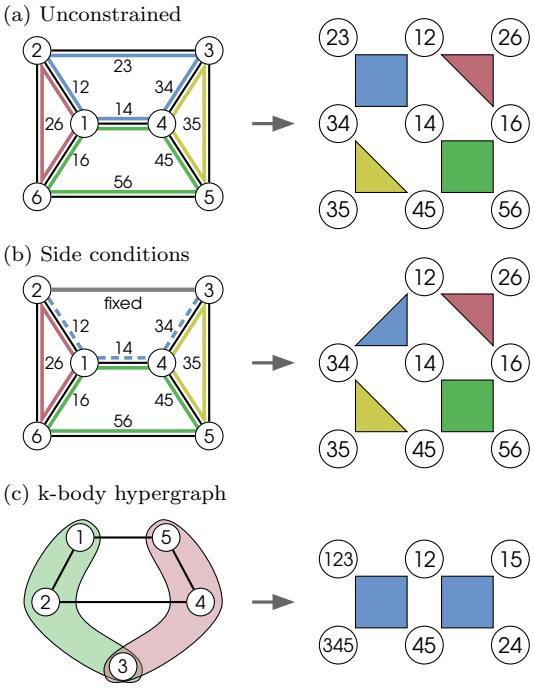

Figure 1: (a) An unconstrained optimization problem is represented by a graph consisting of spins (circles) and edges (black lines). The parity construction is based on finding closed loops of length 3 and 4, e.g., the red triangle connecting spins 1,2 and 6 is a closed loop of length 3. The compiler constructs parity constraints from the closed loops and connects them to a compact quantum chip. In the illustration, colors of the constraints (right) match the colors of the closed loops (left). (b) A side condition to an optimization problem is a hard constraint that has to be satisfied in the solution. As an illustration, we consider the same problem as in (a) but add the side condition that $\sigma_z^{(2)} \sigma_z^{(3)} = +1$. In the parity picture, this can be encoded in *open loops* (blue dashed line). The loop of length 4 is translated by the compiler to a parity constraint between 3 qubits (blue triangle). (c) An optimization problem with higher-order terms is encoded in hypergraphs, e.g., a term $\sigma_z^{(1)} \sigma_z^{(2)} \sigma_z^{(3)}$ is indicated by the green shape. In the parity transformation, higher order terms can be treated exactly the same as edges and thus higher-order terms are encoded without decomposition into a QUBO problem first.

2.1 The embedding and layout maps

It is convenient to think of the transformation, from logical problem to chip layout, as the composition of two maps, which we will call the *embedding* and *layout*

maps. These are maps between *optimization problems*, and thus must preserve the algebra of Pauli Z operators, while mapping states between the following Hilbert spaces:

$$\mathcal{H}_{\text{log}} \overset{\mathcal{P}}{\hookrightarrow} \mathcal{H}_{\text{par}} \overset{\mathcal{L}}{\to} \mathcal{H}_{\text{phys}}. \tag{3}$$

The embedding map $\mathcal{P}$ is a linear map that represents every k-fold product of logical qubits as a single parity qubit, and takes the logical Hilbert space $\mathcal{H}_{\text{log}}$ to a code subspace in the 'parity Hilbert space' $\mathcal{H}_{\text{par}}$ spanned by those parity qubits. The code subspace can be obtained from a set of consistency conditions acting as projections, which arise from demanding that the algebra of Pauli Z operators be preserved. The layout map $\mathcal{L}$ then places the parity qubits on a physical device by mapping them 1-to-1 onto the physical Hilbert space $\mathcal{H}_{\text{phys}}$ in such a way that the consistency conditions can be realized as local interactions. The first map $\mathcal{P} : \mathcal{H}_{\text{log}} \hookrightarrow \mathcal{H}_{\text{par}}$ is straightforward and can be expressed essentially in closed form. The second map $\mathcal{L} : \mathcal{H}_{\text{par}} \to \mathcal{H}_{\text{phys}}$ is hard to compute and typically requires searching the space of possible qubit layouts. In this section we focus on how the embedding map $\mathcal{P}$ is constructed.

2.2 Construction of the parity map

We first define the parity Hilbert space $\mathcal{H}_{\text{par}}$. Suppose the logical Hilbert space $\mathcal{H}_{\text{log}}$ consists of N qubits. The parity transformation will represent each interaction term of the Hamiltonian as a single parity qubit. There are K such interaction terms, and thus $\mathcal{H}_{\text{par}}$ is a tensor product of K qubits for some $0 < K < 2^N$, depending on which k-body interactions are turned on. We denote Pauli Z operators on the i-th logical qubit by $\sigma_z^{(i)}$ for $i \in 1 \dots N$, and on the a-th parity qubit by $\tilde{\sigma}_z^{(a)}$ for $a \in 1 \dots K$. The parity transformation is then given by a collection of mapping relations of the form

$$\sigma_z^{(i_1)} \sigma_z^{(i_2)} \dots \sigma_z^{(i_k)} \mapsto \tilde{\sigma}_z^{(a)}, \tag{4}$$

which maps the k-fold product of logical qubits $i_1, i_2, \dots i_k$ onto a single qubit a in $\mathcal{H}_{\text{par}}$. However, the $\tilde{\sigma}_z^{(a)}$ are a new set of Pauli Z operators and do not respect the algebra implied by the original $\sigma_z^{(i)}$. In order to preserve the Pauli Z algebra, one must impose a set of consistency conditions, which can be obtained from cyclic products of the $\sigma_z^{(i)}$ of the form

$$\sigma_z^{(i_1)} \sigma_z^{(i_2)} \dots \sigma_z^{(i_m)} = 1, \tag{5}$$

where each logical qubit index appears an even number of times. Such cyclic conditions are generalized closed cycles, that correspond to cycles in the logical graph in the original LHZ construction. In both the original case and the generic case with k-body interactions, one obtains *parity projection conditions* on $\mathcal{H}_{\text{par}}$,

$$\tilde{\sigma}_z^{(a_1)} \tilde{\sigma}_z^{(a_2)} \dots \tilde{\sigma}_z^{(a_n)} \, |\tilde{\psi}\rangle = |\tilde{\psi}\rangle \,, \tag{6}$$

for every combination of parity qubits whose preimage along the parity map (4) would give a cyclic product (5). The Pauli Z algebra of $\mathcal{H}_{\text{log}}$ is then preserved along the subspace of $\mathcal{H}_{\text{par}}$ satisfying (6), which is effectively the code subspace. Let us illustrate this using the example in Fig. 1(c), given by the Hamiltonian

$$\begin{aligned} H = {} & J_{12}\, \sigma_z^{(1)} \sigma_z^{(2)} + J_{15}\, \sigma_z^{(1)} \sigma_z^{(5)} \\ & + J_{24}\, \sigma_z^{(2)} \sigma_z^{(4)} + J_{45}\, \sigma_z^{(4)} \sigma_z^{(5)} \\ & + J_{123}\, \sigma_z^{(1)} \sigma_z^{(2)} \sigma_z^{(3)} + J_{345}\, \sigma_z^{(3)} \sigma_z^{(4)} \sigma_z^{(5)}. \end{aligned} \tag{7}$$

If we label the interaction terms by the combination of logical qubits to which they correspond, then we can write the parity transformation as

$$\begin{aligned} \sigma_z^{(1)} \sigma_z^{(2)} &\mapsto \tilde{\sigma}_z^{(1,2)}, & \sigma_z^{(1)} \sigma_z^{(5)} &\mapsto \tilde{\sigma}_z^{(1,5)}, \\ \sigma_z^{(2)} \sigma_z^{(4)} &\mapsto \tilde{\sigma}_z^{(2,4)}, & \sigma_z^{(4)} \sigma_z^{(5)} &\mapsto \tilde{\sigma}_z^{(4,5)}, \\ \sigma_z^{(1)} \sigma_z^{(2)} \sigma_z^{(3)} &\mapsto \tilde{\sigma}_z^{(1,2,3)}, & \sigma_z^{(3)} \sigma_z^{(4)} \sigma_z^{(5)} &\mapsto \tilde{\sigma}_z^{(3,4,5)}, \end{aligned} \tag{8}$$

and one can observe that certain cyclic conditions must hold on the code subspace, such as

$$\tilde{\sigma}_z^{(1,2)} \tilde{\sigma}_z^{(4,5)} \tilde{\sigma}_z^{(1,2,3)} \tilde{\sigma}_z^{(3,4,5)} \, |\tilde{\psi}\rangle = |\tilde{\psi}\rangle \,, \tag{9}$$

which corresponds, in the logical qubits, to the product

$$\sigma_z^{(1)} \sigma_z^{(2)} \sigma_z^{(4)} \sigma_z^{(5)} \sigma_z^{(1)} \sigma_z^{(2)} \sigma_z^{(3)} \sigma_z^{(3)} \sigma_z^{(4)} \sigma_z^{(5)} = 1. \tag{10}$$

One can in principle search for such cyclic products and construct all possible projection conditions in order to define the code subspace in $\mathcal{H}_{\text{par}}$. Once the parity map and projection conditions are constructed, the layout map $\mathcal{L} : \mathcal{H}_{\text{par}} \to \mathcal{H}_{\text{phys}}$ must place all of the parity qubits $\tilde{\sigma}_z^{(a)}$ on the physical device in such a way that the projection conditions in (6) can be realized. In Fig. 1 (c), one can see that the projection (9) has been realized as a square plaquette. However, rather than conducting a search through the logical (hyper)graph to find cyclic products, we find it useful to take a more algebraic approach using the language of classical linear codes. The parity map and projection conditions can be packaged into a *generator matrix* $\mathbf{G}$ and a *parity check matrix* $\mathbf{P}$, which can be manipulated using standard tools of linear algebra, as we now describe.

2.3 Generator and parity check matrices

For an optimization problem with N logical qubits and K interaction terms in the Hamiltonian (whether they be single-body, two-body, or generic k-body interactions), one can describe the code subspace in $\mathcal{H}_{\text{par}}$ via an $N \times K$ *generator matrix* $\mathbf{G}$ which maps

logical bit-string $\mathbf{v}$ into encoded bit-string $\mathbf{w}$ via right-multiplication:

$$\mathbf{w} = \mathbf{v}\mathbf{G}. \tag{11}$$

All linear algebra here should be understood modulo 2; in the bit-strings $\mathbf{w}$ and $\mathbf{v}$, a 0 represents the spin eigenstate $|0\rangle$ (with σ_z eigenvalue $+1$), whereas a 1 represents $|1\rangle$ (with σ_z eigenvalue -1). To construct $\mathbf{G}$ from the mapping in (4), let the a-th column have a 1 in positions $i_1, i_2, \ldots i_k$, and a 0 elsewhere. In the example problem (7), one has $N = 5$ logical qubits and $K = 6$ interaction terms. Each interaction term corresponds to a column of $\mathbf{G}$, which gives us:

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{12}$$

Given the generator matrix, we must then construct the *parity check matrix* $\mathbf{P}$ which effectively encapsulates all the projection conditions. The code subspace in $\mathcal{H}_{\text{par}}$ is defined by the set of all bit-strings $\mathbf{w}$ which satisfy

$$\mathbf{w}\mathbf{P}^\top = 0. \tag{13}$$

Each row of $\mathbf{P}$ represents a single projection condition acting on $\mathcal{H}_{\text{par}}$. One can determine candidate rows of $\mathbf{P}$ by searching for combinations of columns of $\mathbf{G}$ which sum to 0 modulo 2; a row then has a 1 in the positions of the columns participating in a sum, and a 0 elsewhere. From the generator matrix in (12), we can determine

$$\mathbf{P} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}. \tag{14}$$

More generally, the check matrix is the matrix of maximal rank which satisfies

$$\mathbf{G}\mathbf{P}^\top = 0, \tag{15}$$

that is, the rows of $\mathbf{P}$ constitute a basis of the null space of $\mathbf{G}$. Any such basis gives a valid $\mathbf{P}$, and all such bases are related by elementary row operations. The number of rows of $\mathbf{P}$ is always at least $K - N$. In this example, it is $K - N + 1$ because there is a degeneracy in the Hamiltonian (7) (it is invariant under the simultaneous spin flip of qubits 1, 2, 4, 5). For a Hamiltonian with D degeneracies, the number of rows of $\mathbf{P}$ is $K - N + D$.

The check matrix $\mathbf{P}$ gives a complete basis of projection conditions needed to define the code subspace of $\mathcal{H}_{\text{par}}$ on which the mapping relations (4) are consistent. Each projection condition must be realized on the physical device as a constraint coupling, and it will be the task of the layout mapping $\mathcal{L} : \mathcal{H}_{\text{par}} \to \mathcal{H}_{\text{phys}}$ to arrange the physical qubits in such a way that every projection condition can be realized (either by a 3- or 4-body plaquette coupling, or a chain of CNOTs).

2.4 Readout and decoding

Having mapped the original optimization problem onto the set of parity qubits, one must be able to undo this map in order to interpret solutions. Let us first consider the case where all constraints are satisfied. The $N \times K$ *decoding matrix* $\mathbf{D}$ maps encoded bit-strings $\mathbf{w}$ back into logical bit-strings $\mathbf{v}$:

$$\mathbf{w}\mathbf{D}^\top = \mathbf{v}. \tag{16}$$

To find $\mathbf{D}$, we first observe that valid encoded bit-strings always satisfy the projection $\mathbf{w} = \mathbf{w}(1 - \mathbf{P}^\top\mathbf{P})$ as a consequence of (13). Acting with $\mathbf{G}$ on (16) from the right, and using (11) to eliminate $\mathbf{w}$, we obtain

$$(1 - \mathbf{G}\mathbf{D}^\top)\mathbf{G}(1 - \mathbf{P}^\top\mathbf{P}) = 0, \tag{17}$$

which means that $\mathbf{D}^\top$ is a right-pseudoinverse of $\mathbf{G}$, on the subspace projected out by $\mathbf{P}$. That is, given any $\mathbf{D}$ which solves (17), one can obtain another valid solution by adding any row of $\mathbf{P}$ to any row of $\mathbf{D}$. Any such $\mathbf{D}$ is a valid decoding matrix. For example, corresponding to the generator matrix in (12), one can find

$$\mathbf{D} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \tag{18}$$

however, another equivalent solution is

$$\mathbf{D} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{19}$$

which can be obtained by adding the second row of (14) to the first row of (18). They are equivalent decoding matrices because of the algebraic relationships satisfied by valid encoded bit-strings in (13).

The decoding matrix $\mathbf{D}$ informs us how to interpret any state in $\mathcal{H}_{\text{par}}$ which lies in the code subspace defined by the projection conditions. However, in experimental implementations of annealing and digital optimization processes, the quantum state of the physical system may be driven out of the code subspace (either explicitly by the driver Hamiltonian, or by the accumulation of errors), and thus some error-correction scheme must be used before $\mathbf{D}$ can be applied. In this case, the non-uniqueness of $\mathbf{D}$ becomes important, as it implies that there is no uniquely 'correct' logical configuration corresponding to a state in $\mathcal{H}_{\text{par}}$ that fails to satisfy the projections (6). The error-correction process is effectively a method of deciding how to project a generic state in $\mathcal{H}_{\text{par}}$ onto the code subspace, where $\mathbf{D}$ can be applied.

One useful method of making such decisions is belief propagation, as proposed in Ref. [30]. An alternative (perhaps less useful) way is to read out only a minimal subset of parity qubits which are sufficient to reconstruct a logical state, effectively ignoring parity checks.

A simple error-correction scheme was proposed in [29] that builds upon this 'minimal readout' strategy to construct a more robust model. Instead of taking *one* minimal readout, several different ones are taken, on different subsets of qubits, and given a majority vote to determine the most likely logical state. In the original LHZ construction (for 2-body all-to-all problems), a minimal readout subset corresponds precisely to a spanning tree in the logical graph. Thus one can either enumerate *all* such spanning trees and take this majority vote, or draw only a limited number of them from some distribution.

Although it is difficult to generalize the notion of a 'spanning tree' to the hypergraph that represents a problem with k-body interactions, it is straightforward to generalize this notion of 'majority vote between minimal readout subsets'. Once the full *layout* is known (for example, in Fig. 2), one can construct a minimal readout set as follows: Begin at any arbitrary plaquette in the layout. Assign all but one of its qubits to the minimal readout set, and mark them 'read out'. Since these qubits are to be read out, their values are known. The remaining qubit in this plaquette is unknown, but we assume the constraint represented by this plaquette holds, and thus the value of this qubit can be deduced from the read out qubits; mark it 'determined'. To continue constructing the readout set, choose any plaquette which is adjacent to those plaquettes already visited; that plaquette will have some qubits 'read out', some qubits 'determined', and some qubits unmarked. Of the unmarked qubits, mark all but one of them as 'read out', and the final one as 'determined'. Continue in this fashion until all qubits have been marked. Then the ones marked 'read out' constitute a minimal readout subset.

3 Constrained optimization problems

For some optimization problems, one may wish to impose additional conditions on the possible solutions, beyond the Hamiltonian (1). Such *constrained optimization problems* have a number of constraints which act in the logical spin space $\mathcal{H}_{\log}$. These can be product constraints $\sigma_1\sigma_2\sigma_3 = 1$ or sum constraints $\sigma_1 + \sigma_2 + \sigma_3 = 1$ (or generically, polynomial constraints, which combine both principles).

3.1 Product constraints

Product constraints can be realized as projections in the logical Hilbert space $\mathcal{H}_{\log}$. A collection of constraints of the form

$$\sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_k}\,|\psi\rangle = |\psi\rangle \tag{20}$$

can be represented, in our linear-algebra language, by imposing a condition on logical bit-strings:

$$\mathbf{v}\mathbf{C}^\top = 0, \tag{21}$$

where the *constraint matrix* $\mathbf{C}$ has N columns and any number of rows; in a given row, a 1 in the i-th column indicates the presence of σ_i in a product constraint. The rest of the algebraic structure of the linear code goes through as described above, but with the caveat that the space of logical bit-strings has been restricted to a subspace in $\mathcal{H}_{\log}$ given by the projection

$$\mathbf{v}(1 - \mathbf{C}^\top\mathbf{C}) = \mathbf{v}. \tag{22}$$

The parity check matrix $\mathbf{P}$ is then given by the matrix of maximal rank which satisfies

$$(1 - \mathbf{C}^\top\mathbf{C})\mathbf{G}\mathbf{P}^\top = 0, \tag{23}$$

and the decoding matrix $\mathbf{D}$ is any simultaneous solution of

$$(1 - \mathbf{C}^\top\mathbf{C})(1 - \mathbf{G}\mathbf{D}^\top)\mathbf{G}(1 - \mathbf{P}^\top\mathbf{P}) = 0, \tag{24}$$

$$(1 - \mathbf{P}^\top\mathbf{P})\mathbf{D}^\top\mathbf{C}^\top = 0. \tag{25}$$

That is, $\mathbf{D}^\top$ is a right-pseudoinverse of $\mathbf{G}$ on the constrained subspace of logical bit-strings, and also satisfies the constraint condition itself for all valid encoded bit-strings $\mathbf{w} = \mathbf{w}(1 - \mathbf{P}^\top\mathbf{P})$.

For our example Hamiltonian (7), suppose we impose the constraint $\sigma_1\sigma_2\sigma_4 = 1$, which is sufficient to fix the spin-flip degeneracy in qubits $1, 2, 4, 5$. This is represented by the constraint matrix

$$\mathbf{C} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \end{pmatrix}. \tag{26}$$

The parity check matrix $\mathbf{P}$ is unchanged, since this constraint only fixes a degeneracy and has no effect on the code subspace in $\mathcal{H}_{\text{par}}$. The decoding matrix becomes

$$\mathbf{D} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}, \tag{27}$$

which can be obtained from (18) via elementary row operations. In Ref. [27] we show how to encode the more general sum constraints.

4 Compilation

Having defined the mapping from logical qubits to qubits in parity space $\mathcal{H}_{\text{par}}$ and obtained the necessary basis of projection conditions encoded in the

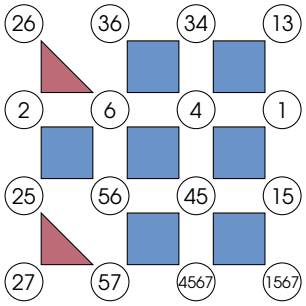

Figure 2: Example of a plaquette compilation from a logical Hamiltonian on 7 logical qubits with 16 interactions. Because of the four-body interactions, this problem cannot be directly encoded using the LHZ mapping. The physical qubits are labeled with the combination of logical qubits that participate in the corresponding interaction. The code subspace is obtained from a combination of projectors of length 4 (blue) and length 3 (red).

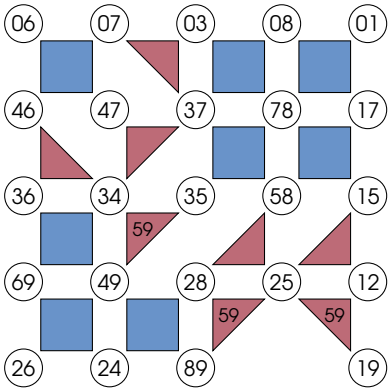

Figure 3: Example plaquette compilation with side conditions. The original Hamiltonian has 10 logical qubits appearing in 24 interactions, which requires 45 physical qubits in the LHZ mapping. The physical qubits are labeled by the combination of logical qubits in the corresponding interaction term. The constraint $\sigma_z^{(5)}\sigma_z^{(9)} = 1$ is imposed in the logical problem. Elementary row operations on $\mathbf{P}$ have been used to subsume this constraint into three projection conditions, $\tilde{\sigma}_z^{(3,4)}\tilde{\sigma}_z^{(3,5)}\tilde{\sigma}_z^{(4,9)} = 1$, $\tilde{\sigma}_z^{(2,8)}\tilde{\sigma}_z^{(2,5)}\tilde{\sigma}_z^{(8,9)} = 1$, and $\tilde{\sigma}_z^{(2,5)}\tilde{\sigma}_z^{(1,2)}\tilde{\sigma}_z^{(1,9)} = 1$.

parity matrix $\mathbf{P}$, we now discuss the second step of the compilation process, the layout map $\mathcal{L} : \mathcal{H}_{\text{par}} \to \mathcal{H}_{\text{phys}}$, which implements the various code subspace projections on a physical device. The qubits of $\mathcal{H}_{\text{par}}$ must be mapped 1-to-1 onto physical qubits, in such a way that the projection conditions $\mathbf{P}$ can be realized by physical couplings. The physical qubits and available couplings are at fixed locations on the device, and thus computing a valid layout $\mathcal{L}$ is a problem akin to graph isomorphism, and requires some sort of search algorithm. In this section we will discuss some techniques by which the parity compiler achieves such layouts.

There are two main types of devices we consider, based on how the projection conditions of $\mathbf{P}$ can be realized: *plaquette devices* and *CNOT devices*.

4.1 Compilation via plaquettes

We will first consider plaquette compilation. On a plaquette device there exist a number of 3- and 4-body couplers which can constrain the parity of spins situated in a square or triangle. The ideal device which we consider here has a 4-body coupler available on every grid square, which can be (optionally) chosen to constrain only 3 corners of the square, making it serve as a 3-body coupler when needed.

Plaquette compilation of Hamiltonians with higher-order k-body terms (HCBO) as in (1) is a straightforward generalization of the original LHZ proposal for Hamiltonians with 2-body interactions. The projections onto the code subspace are realized by square or triangle couplers in a square lattice, as in Fig. 2. These couplers add terms to the physical Hamiltonian of the form

$$-C_{ijk(\ell)}\,\tilde{\sigma}_z^{(i)}\tilde{\sigma}_z^{(j)}\tilde{\sigma}_z^{(k)}(\tilde{\sigma}_z^{(\ell)}), \tag{28}$$

where C is some coupling strength chosen to be sufficiently large, so that the physical ground state is one where the product of spins going around every plaquette is $+1$.

Using plaquettes on a square lattice, it is only possible to realize projectors with 3 or 4 σ_z operators. This means that the rows of the parity check matrix $\mathbf{P}$ must have Hamming weight 3 or 4. If this is not the case, then there are a few options. First, one may use elementary row operations on $\mathbf{P}$ to transform it into a matrix whose rows have Hamming weight 3 or 4. If this is not possible, then one can instead introduce ancilla qubits in order to decompose longer projectors into shorter ones. For example, the projector

$$\tilde{\sigma}_z^{(1)}\tilde{\sigma}_z^{(2)}\tilde{\sigma}_z^{(3)}\tilde{\sigma}_z^{(4)}\tilde{\sigma}_z^{(5)} = 1 \tag{29}$$

can be realized as the combination of two projectors

$$\tilde{\sigma}_z^{(1)}\tilde{\sigma}_z^{(2)}\tilde{\sigma}_z^{(6)} = 1, \qquad \sigma_z^{(3)}\tilde{\sigma}_z^{(4)}\tilde{\sigma}_z^{(5)}\tilde{\sigma}_z^{(6)} = 1, \tag{30}$$

where 6 is a (physical) ancilla qubit.

4.1.1 Implementing side conditions

We have discussed how constrained optimization problems with product constraints can be subsumed into the same linear-algebraic framework for parity compilation. Essentially the parity check matrix $\mathbf{P}$ is constructed such that it contains all the necessary information already. In Fig. 3 we give an example.

4.1.2 Choosing ancillas

Occasionally it is necessary to add ancilla qubits in order to compile certain problems. An ancilla qubit in this context is any physical qubit which does not correspond to a term in the original Hamiltonian; its

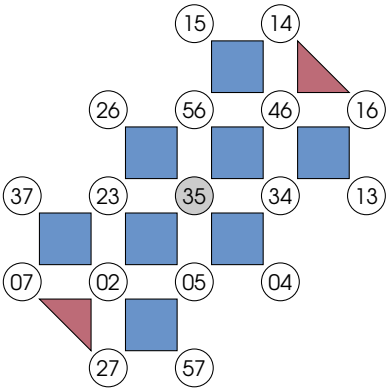

Figure 4: Plaquette compilation using a pre-constructed ancilla, shown in grey. The ancilla qubit $\sigma_z^{(3,5)}$ makes it possible to lay out the constraints. Note that here 17 physical qubits are required compared to the 28 needed in the LHZ mapping.

only purpose is to assist us in implementing the parity projections on the other qubits.

There are essentially two situations in which an ancilla may need to be added. The first situation is one we have already mentioned: when the rows of $\mathbf{P}$ have Hamming weight greater than 4, and this cannot be remedied by elementary row operations. In this case, an ancilla must be created to break up a long projector into shorter ones. One can make choices about exactly how to do this, since a given long projector might be broken up in many different ways. In some cases, there may be heuristics which prefer one choice over another, such as when a single ancilla may serve to break up several long projectors. An example of a compiled problem using an ancilla qubit is given in Fig. 4.

The second situation for using ancillas is for additional flexibility in the search for a valid layout. For example, one may choose, dynamically during the course of compilation, to break up a length-4 projector by adding an ancilla which splits it into two length-3 projectors. The advantage is that the original 4 physical qubits no longer have to be adjacent in the layout. Such 'dynamical ancillas' allow one to find layouts that might otherwise be impossible.

In general, the main drawback of adding ancillas is that for every ancilla, one also must add an extra projection (effectively, the matrix $\mathbf{P}$ grows by one row). So more physical resources (qubits and couplings) are required for a given logical problem size. However, sometimes such ancillas are necessary in order to find a layout for a given problem. They must be chosen judiciously.

4.2 Compilation for digital optimization

Next we will consider compilation to devices whose couplings are CNOT gates acting along the edges of a square lattice, and local Z rotations acting on the vertices. These gates can be used to implement unitary operators corresponding to time evolution under spin parity constraints, suitable for digital optimization algorithms [31–33]. The projection conditions given by the parity matrix $\mathbf{P}$ can be realized with these CNOT gates and Z rotations rather than with 4-body plaquette couplers, as we will describe. This type of compilation is well-suited to devices with an array of CNOT gates (or equivalent), and it allows additional flexibility in constructing the physical layout, which provides further opportunity for optimizations, such as increasing parallelization or minimizing circuit depth.

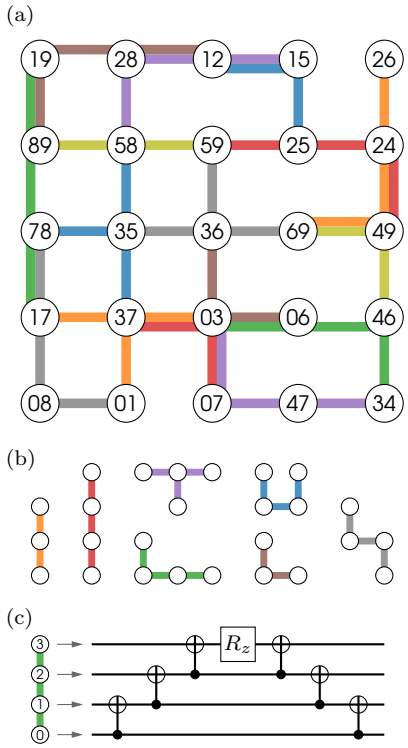

Figure 5: (a) Example of a compilation implementing the code space projectors with CNOT gates. Each coloured path represents a single projector which constrains the physical qubits along that path to have even parity. Implementing this using the LHZ mapping requires 45 physical qubits. (b) All possible shapes of projectors used during the layout process. (c) The decomposition of a projection condition into a chain of CNOT gates and a rotation, implementing the unitary $U = \exp(-i\frac{\alpha}{2}\sigma_z^{(0)}\sigma_z^{(1)}\sigma_z^{(2)}\sigma_z^{(3)})$, where α is the rotation angle of R_z.

An example of a CNOT compilation is given in Fig. 5. Each code space projector is realized by a tree of edges in the physical graph. CNOT gates are executed from the leaves of the tree towards its root, tying a set of qubits together, and a local Z rotation at the end of the path serves to implement the appropriate parity-constraining unitary operation. One can in principle lay out projectors of any length this way, although it is useful to bound the length in order to limit the circuit depth of the chain of CNOTs, and thus one will want to use elementary row opera-

tions on **P** or introduce ancillas, just as in the case of plaquette projectors.

The trees or paths of CNOTs may be laid out in any shape; the only requirement is that all the qubits in a given projector be contiguous in the layout. This high degree of flexibility can greatly aid in finding such layouts. One can also consider imposing additional conditions, such as maximizing the number of CNOTs which can be run in parallel, in order to minimize the total circuit depth.

5 Conclusion

We presented an architecture to encode HCBO problems i.e. optimization problems with higher-order interactions and side conditions to simple square lattices. This is a generalization of the original LHZ mapping in several ways: The main generalization is that from all-to-all models of pair interactions (or k-body terms) to problems with mixtures of k-body terms of different k and less than all-to-all connectivity. The parity mapping translates products of σ_z into single parity qubits. Using the generalized conditions that replace the closed cycles in LHZ, this allows for the encoding of arbitrary mixtures of k-body terms. A main advantage of compilation compared to LHZ is the reduction of the number of qubits from an N^2 scaling to a K scaling, where K is the number of terms in the Hamiltonian, independent of the k-locality of these terms. For digital quantum algorithms, this reduction in qubit overhead again provides an advantage in gate counts, which is amplified by the lack of need for SWAP gates, especially for dense problem graphs with higher-order interactions [28]. In compiled plaquette layouts decomposed into gates for digital devices, the resulting circuits are fully parallelizable and enable algorithms to be implemented with system-size-independent circuit depth.

Another novelty reported here is that side conditions in the form of product constraints can be included without overhead. This is achieved by another generalization of the LHZ cycles, namely the generalization to open cycles. The implementation of more general constraints, i.e. constraints on sums of products in the logical spin variables, is investigated in [27]. Depending on the available hardware and the specific problem such constraints can also be encoded without overhead in required resources.

In combination with a compiler, i.e. classical software that finds the optimal layout of the plaquettes on a chip, the parity mapping is a powerful tool to encode optimization problems with side conditions on current quantum hardware. Taking all these advantages into account, the parity architecture may serve as a blueprint for next generation quantum optimization devices [34, 35].

Acknowledgements - Work at the University of Innsbruck is supported by the European Union program Horizon 2020 under Grants Agreement No. 817482 (PASQuanS), and by the Austrian Science Fund (FWF) through a START grant under Project No. Y1067-N27 and the SFB BeyondC Project No. F7108-N38, the Hauser-Raspe foundation. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0068. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

References

[1] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. "Optimization by simulated annealing". Science **220**, 671–680 (1983).

[2] Andrew Lucas. "Ising formulations of many np problems". Frontiers in Physics **2**, 5 (2014).

[3] J. I. Cirac and P. Zoller. "Quantum computations with cold trapped ions". Phys. Rev. Lett. **74**, 4091–4094 (1995).

[4] Rainer Blatt and Christian F Roos. "Quantum simulations with trapped ions". Nature Physics **8**, 277–284 (2012).

[5] David Kielpinski, Chris Monroe, and David J Wineland. "Architecture for a large-scale ion-trap quantum computer". Nature **417**, 709–711 (2002).

[6] D Jaksch, JI Cirac, P Zoller, *et al.* "Fast quantum gates for neutral atoms". Phys. Rev. Lett. **85**, 2208–2211 (2000).

[7] Loïc Henriet, Lucas Beguin, Adrien Signoles, *et al.* "Quantum computing with neutral atoms". Quantum **4**, 327 (2020).

[8] M. Saffman, T. G. Walker, and K. Mølmer. "Quantum information with rydberg atoms". Rev. Mod. Phys. **82**, 2313–2363 (2010).

[9] Immanuel Bloch, Jean Dalibard, and Wilhelm Zwerger. "Many-body physics with ultracold gases". Rev. Mod. Phys. **80**, 885–964 (2008).

[10] Hannes Bernien, Sylvain Schwartz, Alexander Keesling, *et al.* "Probing many-body dynamics on a 51-atom quantum simulator". Nature **551**, 579–584 (2017).

[11] Jens Koch, Terri M. Yu, Jay Gambetta, *et al.* "Charge-insensitive qubit design derived from the cooper pair box". Phys. Rev. A **76**, 042319 (2007).

[12] Andreas Wallraff, David I Schuster, Alexandre Blais, *et al.* "Strong coupling of a single photon to a superconducting qubit using circuit quantum electrodynamics". Nature **431**, 162–167 (2004).

[13] Mark W Johnson, Mohammad HS Amin, Suzanne Gildert, *et al.* "Quantum annealing with manufactured spins". Nature **473**, 194–198 (2011).

[14] Frank Arute, Kunal Arya, Ryan Babbush, *et al.* "Quantum supremacy using a programmable superconducting processor". Nature **574**, 505–510 (2019).

[15] L Childress, MV Gurudev Dutt, JM Taylor, *et al.* "Coherent dynamics of coupled electron and nuclear spin qubits in diamond". Science **314**, 281–285 (2006).

[16] Jeremy L O'brien, Akira Furusawa, and Jelena Vučković. "Photonic quantum technologies". Nature Photonics **3**, 687–695 (2009).

[17] Xiaogang Qiang, Xiaoqi Zhou, Jianwei Wang, *et al.* "Large-scale silicon quantum photonics implementing arbitrary two-qubit processing". Nature photonics **12**, 534–539 (2018).

[18] G. G. Guerreschi and A. Y. Matsuura. "Qaoa for max-cut requires hundreds of qubits for quantum speed-up". Scientific Reports **9**, 6903 (2019).

[19] Edward Farhi, David Gamarnik, and Sam Gutmann. "The quantum approximate optimization algorithm needs to see the whole graph: Worst case examples" (2020). arXiv:2005.08747.

[20] Tameem Albash and Daniel A. Lidar. "Adiabatic quantum computation". Rev. Mod. Phys. **90**, 015002 (2018).

[21] Tadashi Kadowaki and Hidetoshi Nishimori. "Quantum annealing in the transverse ising model". Phys. Rev. E **58**, 5355–5363 (1998).

[22] Philipp Hauke, Helmut G Katzgraber, Wolfgang Lechner, *et al.* "Perspectives of quantum annealing: methods and implementations". Reports on Progress in Physics **83**, 054401 (2020).

[23] Rongxin Xia, Teng Bian, and Sabre Kais. "Electronic structure calculations and the ising hamiltonian". The Journal of Physical Chemistry B **122**, 3384–3395 (2018).

[24] Alejandro Perdomo-Ortiz, Neil Dickson, Marshall Drew-Brook, *et al.* "Finding low-energy conformations of lattice protein models by quantum annealing". Scientific Reports **2**, 571 (2012).

[25] Baonan Wang, Feng Hu, Haonan Yao, and Chao Wang. "Prime factorization algorithm based on parameter optimization of ising model". Scientific Reports **10**, 7106 (2020).

[26] Román Orús, Samuel Mugel, and Enrique Lizaso. "Forecasting financial crashes with quantum computing". Phys. Rev. A **99**, 060301 (2019).

[27] Maike Drieb-Schön, Younes Javanmard, Kilian Ender, and Wolfgang Lechner. "Parity quantum optimization: Encoding constraints" (2021). arXiv:2105.06235.

[28] Michael Fellner, Kilian Ender, Roeland ter Hoeven, and Wolfgang Lechner. "Parity quantum optimization: Benchmarks" (2021). arXiv:2105.06240.

[29] Wolfgang Lechner, Philipp Hauke, and Peter Zoller. "A quantum annealing architecture with all-to-all connectivity from local interactions". Science Advances **1** (2015).

[30] Fernando Pastawski and John Preskill. "Error correction for encoded quantum annealing". Phys. Rev. A **93** (2016).

[31] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm" (2014). arXiv:1411.4028.

[32] Wolfgang Lechner. "Quantum approximate optimization with parallelizable gates". IEEE Transactions on Quantum Engineering **1**, 1–6 (2020).

[33] Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, *et al.* "Quantum optimization using variational algorithms on near-term quantum devices". Quantum Science and Technology **3**, 030503 (2018).

[34] John Preskill. "Quantum Computing in the NISQ era and beyond". Quantum **2**, 79 (2018).

[35] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, *et al.* "Noisy intermediate-scale quantum algorithms". Rev. Mod. Phys. **94**, 015004 (2022).