Determining the Minimal Number of SWAP Gates for Multi-dimensional Nearest Neighbor Quantum Circuits

Aaron Lye[1] Robert Wille[1,2] Rolf Drechsler[1,2]

[1]Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
[2]Cyber Physical Systems, DFKI GmbH, 28359 Bremen, Germany
{lye,rwille,drechsle}@informatik.uni-bremen.de

Abstract

Motivated by the promises of significant speed-ups for certain problems, quantum computing received significant attention in the past. While much progress has been made in the development of synthesis methods for quantum circuits, new physical developments constantly lead to new constraints to be addressed. The limited interaction distance between the respective qubits (i.e. nearest neighbor optimization) has already been considered intensely. But with the emerge of multi-dimensional quantum architectures, new physical requirements came up for which only a few automatic synthesis solutions exist yet – all of them of heuristic nature. In this work, we propose an exact scheme for nearest neighbor optimization in multi-dimensional quantum circuits. Although the complexity of the problem is a serious obstacle, our experimental evaluation shows that the proposed solution is sufficient to allow for a qualitative evaluation of the respective optimization steps. Besides that, this enabled an exact comparison to heuristical results for the first time.

I. Introduction

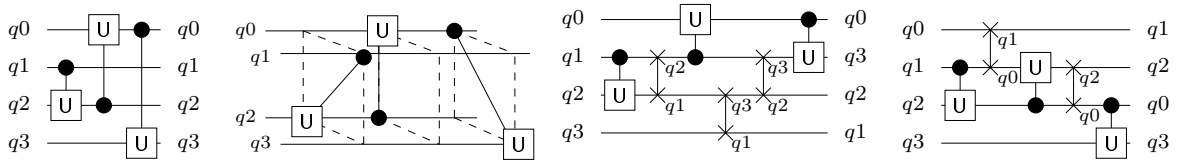
Quantum computing [1] allows for solving many relevant problems in significantly less complexity. This is achieved by exploiting certain quantum mechanical phenomena. For example, information can not only be represented by the conventional basis states 0 and 1, but also by their superposition. Additional properties such as entanglement can be utilized to solve problems like factorisation [2] or database search [3] significantly faster than with conventional concepts. Motivated by these developments, researchers and engineers started to actively consider the logic synthesis of the corresponding circuits.

First netlists of the respective quantum circuits have thereby been developed by hand. But in order to design even more complex quantum circuits, automatic methods for computer-aided design and synthesis are required. Accordingly, efficent quantum circuit design became an active field of research. Therefore, several technological constraints have to be considered during the design process. Among them, the limited interaction distance between gate qubits is one of the most common ones. Here, it is required that computations are only to be performed between adjacent, i.e. nearest neighbor, qubits.

An established design scheme to address this constraint is to apply a post-synthesis optimization to the quantum circuits which are usually composed of 1- and 2-qubit gates. The main idea of most of these approaches is to add so called *SWAP gates* into the circuit structure in order to move the respective gate connections together until they become adjacent. However, the precise fashion how these SWAP gate insertions are conducted has a significant effect on the overall costs of the resulting circuit. In particular, reordering the qubit positions or considering SWAP gate insertion not only locally for each single gate but for the whole cascade may reduce the costs significantly. Hence, several approaches on determining good SWAP gate insertions have been proposed in the past [4, 5, 6, 7, 8, 9].

So far, these approaches assumed quantum circuit realizations in one-dimensional architectures. But recently, physical implementations based on multi-dimensional architectures have gained interest and are seen as the more suitable physical solution [10, 11, 12]. Here, qubits are not aligned next to each other, but e.g. in a 2D structure. First physical realizations based on photonics [13], superconductors [13], quantum dots [14], and neutral atoms [15] have been shown very promising. Nevertheless, the development of respective synthesis solutions ensuring nearest neighbor-compliance for these architectures are just at the beginning. To the best of our knowledge, only hand-made solutions such as [16] or the approach recently proposed in [17] exists yet. However, these solutions are of heuristic nature and, hence, no exact results on 2D and multi-dimensional nearest neighbor quantum circuits have been obtained thus far. That is, exactly evaluating the performance of nearest neighbor optimization for multi-dimensional quantum architectures was not possible until today.

In this paper, we aim for such an exact evaluation. We observe that, in contrast to nearest neighbor optimization for 1D architectures, determining the minimal number of SWAP gates for multi-dimensional quantum architectures requires several complex sub-problems to be solved. Consequently, we propose a solution composed of three steps – each of them tackling a separate sub-problem. Experiments confirm the computational complexity of the considered problem, but also show that – using the proposed solutions – it is possible to determine exact results for circuits of up to six qubits. This is sufficient to allow for a qualitative evaluation of the respective optimization steps to be performed in order to generate nearest-neighbor-compliant quantum circuits for multi-dimensional architectures. At the same time, this enables, for the first time, an exact comparison to results which have heuristically been generated before.



(a) 1D circuit (b) 2D circuit (c) Nearest neighbor 1D circuit (d) Cheaper NN 1D circuit

Fig. 1. Multi-dimensional quantum circuits

The remainder of this paper is structured as follows: In order to keep this paper self-contained, Section II introduces the basics of quantum circuits and nearest neighbor architectures. Section III introduces the problem considered in this work in more detail and proposes the main idea of our solution. Afterwards, the solution is comprehensively described in Section IV. Section V shows the application and discusses the achieved results in comparison to previously obtained results. Finally, the paper is concluded in Section VI.

II. Nearest Neighbor Quantum Circuits

Quantum computing relies on manipulating quantum bits (*qubits*) rather than conventional bits. A qubit may assume any state represented by the linear superposition of the basis states $|0\rangle$ and $|1\rangle$. The manipulations are performed by applying specific unitary operations U. Commonly used quantum operations include the Hadamard operation H (setting a qubit into superposition), the phase shift operation S, as well as the NOT operation X. Details on these operations are not relevant in the remainder of this work, but can be found e.g. in [1]. The application of a unitary operation is eventually represented by means of quantum gates, i.e. an *n-qubit quantum gate* applies a $2^n \times 2^n$ unitary matrix to the corresponding qubits. This leads to the following definition of a quantum circuit used in this work.

Definition 1. *A quantum circuit is denoted by the cascade $G = g_1 g_2 \dots g_{|G|}$ (in figures drawn from left to right), where $|G|$ denotes the total number of gates. The number of qubits is denoted by n. Usually, quantum circuits are composed of* unary gates *simply applying the respective unitary operation on a single qubit or* controlled quantum gates *over two qubits. The* costs *of a quantum circuit are defined by the number $|G|$ of gates.*

Originally, qubits in a quantum circuit have been arranged in a 1-dimensional (i.e. linear) fashion where each qubit is placed next to each other. Fig. 1(a) shows an example of such a circuit. However, recent technological developments (e.g. [10, 11, 12]) also lead to the consideration of 2-dimensional arrangements where qubits are placed according to a grid-structure. In this case, the circuit from Fig. 1(a) would be realized as sketched in Fig. 1(b). Such arrangements can accordingly been extended to higher dimensions eventually leading to *multi-dimensional quantum circuits*.

Finally, technological constraints for certain technologies (see e.g. [12, 18, 19]) limit the interaction distance between the qubits and, hence, only allow the application of gates between adjacent (i.e. nearest neighbor) qubits. For the 1D circuit depicted in Fig. 1(a) and the 2D circuit depicted Fig. 1(b), this only holds for gate g_1 and g_2, respectively. All other gates have to be made nearest neighbor-compliant. To this end, so called SWAP gates can be utilized.

Definition 2. *A* SWAP *gate over two qubits q_i, q_j transforms $(q_0, \dots, q_{i-1}, q_i, q_{i+1} \dots, q_{j-1}, q_j, q_{j+1} \dots, q_{n-1})$ to $(q_0, \dots, q_{i-1}, q_j, q_{i+1} \dots, q_{j-1}, q_i, q_{j+1} \dots, q_{n-1})$, i.e. simply swaps the value of the two qubits q_i and q_j.*

Nearest neighbor-compliance of a quantum circuit can be achieved by applying a cascade of adjacent SWAP gates in front of each gate g with non-adjacent qubits. By this, the respective qubits are shifted together until they are adjacent. Fig. 1(c) exemplary shows the circuit previously considered in Fig. 1(a) which has been made nearest neighbor compliant by inserting additional SWAP gates (SWAP gate connections are denoted by $\times$). In a similar fashion, this can be applied to the 2D circuit from Fig. 1(b) as well.

III. Considered Problem and Proposed Solution

Ensuring nearest neighbor compliance by inserting SWAP gates has heavily been considered in the past and is an established procedure to satisfy the underlying technological constraints. The main objective is thereby to keep the number of SWAP gates to be inserted as small as possible. This section reviews the respective related work and, based on that, states the research question considered in this work. Afterwards, the proposed steps to solve the considered problem are outlined.

A. Related Work and Considered Problem

Obviously, the fashion in which SWAP gates are inserted has a significant effect on the required number of insertions. For example, the insertion of SWAP gates as shown in Fig. 1(d) leads to a much cheaper nearest neighbor-compliant circuit compared to the solution previously shown in Fig. 1(c). Accordingly, different approaches for optimizing the insertion of SWAP gates have been proposed in the recent past.

The majority of them focused on 1D quantum circuits and applied strategies such as the re-ordering of qubit positions [5], window-based heuristics [7], or mapping the problem to a corresponding graph arrangement problem [4]. While these approaches lead to heuristic solutions only, an exact approach guaranteeing the minimal number of SWAP insertions has recently been proposed in [8, 9]. In [6], another exact approach has been proposed which additionally allows for changing the order of the gates so that the results obtained there are not directly comparable to the other solutions.

In contrast, nearest neighbor optimization for 2D (or even multi-dimensional) quantum circuits is just at the beginning: Although manually derived nearest neighbor-compliant 2D realizations for certain building blocks such as an adder, e.g. in [16], have been presented, automatic approaches for SWAP gate insertion are hardly available yet. To the best of our knowledge, only the approach recently proposed in [17] exists. This, however, only generates heuristic solutions. Until today, no exact approach and, hence, no exact results on 2D and multi-dimensional nearest neighbor quantum circuits exists yet. This is crucial since clear results e.g. on the number of needed SWAP gates or the best possible qubit arrangement cannot be derived from heuristic results.

Motivated by this, we consider the following problem in this work: *How to determine the* minimal *number of SWAP gates to be inserted to make a 2D or even a generic, i.e.* multi-dimensional*, quantum circuit nearest neighbor-compliant.*

B. Proposed Solution

Determining the (minimal) number of SWAP gate insertions in 1D quantum circuits basically focused on where and how many SWAP gates have to be added into an existing circuit structure. Considering multi-dimensional quantum circuits, further issues need to be addressed. For example, the precise configuration of the circuit is not fix: A 1D circuit with e.g. 7 qubits always have those seven qubits arranged next to each other; in a 2D quantum circuit, they may be arranged onto a 2×4- or even 3×3-grid. Multi-dimensional quantum circuits allow for many further possibilities. This obviously has an effect on which qubits are adjacent and which are not. Moreover, even the number of SWAP gates needed to establish a certain qubit permutation onto the circuit (e.g. in order to make a non-adjacent gate nearest neighbor-compliant) significantly depends on this. Determining this number is a non-trivial task.

In order to address all these issues, we propose a solution composed of the following steps:

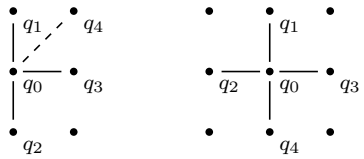
1. Determine the precise configuration of the considered quantum circuit, i.e. its dimensions for a given number of qubits.

2. Determine a cost function providing the *minimal* number of SWAP gates needed to realize an arbitrary permutation onto the given circuit configuration.

3. Determine the minimal SWAP gate insertions based on the given circuit configuration and its cost function.

IV. Implementation

This section describes the proposed solution and the applied procedures to each of the steps mentioned above in detail. For sake of clarity, all issues are mostly discussed and illustrated by means of 2D quantum circuits. However, all concepts can accordingly be extended to multi-dimensional quantum circuits.



(a) 2×3-grid (b) 3×3-grid

Fig. 2. Determine the precise quantum circuit configuration

A. Determine the Precise Quantum Circuit Configuration

For a given number of qubits, multi-dimensional quantum circuits allow for several possible configurations to be considered. The respective choice has thereby a significant effect on the number of needed SWAP gates but also on the number of *garbage*, i.e. unused, qubit positions.

Example 1. *Consider a quantum circuit over five qubits $q_0,\dots,q_4$ to be realized in a 2D architecture and with interactions between q_0 and each of $q_1,\dots,q_4$. The smallest possible 2D configuration would require a 2×3-grid*[1]*. As sketched in Fig. 2(a), the best possible qubit-placement indeed would keep the number of garbage positions minimal (just one), but additionally requires an additional SWAP gate for q_0 and q_4. In contrast, a 3×3-grid would allow for a qubit-placement with no need for any additional SWAP gates but eventually lead to four garbage qubit positions (as sketched in Fig. 2(b)).*

In general, researchers and designers aim for keeping the number of garbage qubits as small as possible [20]. At the same time, also efforts have been made to explicitly exploit those [21, 22]. Multi-dimensional nearest neighbor quantum circuits provide another argument for being more flexible with the "keep the number of garbage qubits as small as possible"-design rule. Eventually, the designer has to trade-off the respective criteria. In the following, we aim for determining the minimal number of SWAP gates for quantum circuit configurations with the minimal number of garbage qubit positions. However, the approach is also applicable for configurations which are larger than necessary.

B. Costs of Establishing an Arbitrary Permutation

In order to globally determine the minimal number of SWAP gates needed to make an arbitrary quantum circuit nearest neighbor-compliant, it has to be known how many SWAP gates are required to establish an arbitrary permutation of qubit positions in that circuit. For 1D quantum circuits, this can been obtained in linear time using *inversion vectors* [8, 9]. For multi-dimensional circuits, this constitutes a more complex problem. The problem can be formulated by means of adjacent transposition graphs[2].

Definition 3. *Let $G = g_1g_2\dots g_{|G|}$ be a circuit over n qubits. An* adjacent transposition graph *$A = (V, E)$ is a representation of all transpositions which can be realized by nearest neighbor-compliant SWAP gates. The set V of nodes represent all $|V| = n!$ possible permutations while edges represent valid transpositions from one permutation to another.*

[1] In principle, a 1×5-grid may be considered the smallest possible configuration but, however, is considered as a 1D configuration.

[2] Adjacent transposition graphs have previously also been applied in [6] for nearest neighbor optimization of 1D circuits. However, as said above *inversion vectors* are the more efficient solution here.

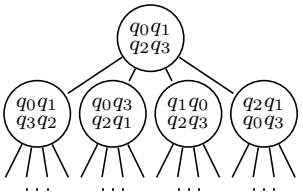


Fig. 3. Adjacent transposition graph

Example 2. *Fig. 3 sketches a part of the transposition graph for a 2D quantum circuit over $n = 4$ qubits. Transition graphs for multi-dimensional quantum circuits can be created by accordingly considering the possible transpositions and the resulting permutations.*

Having such a representation, the minimal number of SWAP gates needed to establish an arbitrary permutation can be obtained by determining a minimal path from the node representing the identity permutation (denoted as v_{src}) to the node representing the desired permutation (denoted as v_{dest}). Inspired by [23], this can be formulated as *Pseudo-Boolean Optimization problem* (PBO problem).

More precisely, for each node $v \in V$ a Boolean variable x_v is introduced representing whether v is included in the optimal path from the source to the destination node, i.e. $x_v = 1$ iff v is in the optimal path. In the same fashion, a Boolean variable x_e is introduced for each edge $e \in E$ representing whether e is included in the optimal path from the source to the destination node, i.e. $x_e = 1$ iff e is in the optimal path.

Then, it has to be constrained that (1) v_{src} and v_{dest} are part of the path, (2) one edge including v_{src} and one including v_{dest} have to be part of the path, and (3) if any other node $v \in V \setminus \{v_{src}, v_{dest}\}$ is part of the path (i.e. iff $x_v = 1$), then two other edges incident to v (an incoming one and an outcoming one) have to be part of the path as well. Minimality of the path is ensured by enforcing the optimization function (4), i.e.

$$x_{v_{src}} \wedge x_{v_{dest}} \tag{1}$$

$$\wedge \sum_{e \in \{(v_{src}, \bullet) \in E\}} x_e = 1 \wedge \sum_{e \in \{(v_{dest}, \bullet) \in E\}} x_e = 1 \tag{2}$$

$$\bigwedge_{v \in V \setminus \{v_{src}, v_{dest}\}} (x_v \Leftrightarrow \sum_{e \in \{(v, \bullet) \in E\}} x_e = 2), \tag{3}$$

$$min : \sum_{e \in E} x_e. \tag{4}$$

Passing this formulation to a state-of-the-art PBO solver [24], a minimal assignment to all x_v- and x_e-variables is derived. From the x_e-variables, the minimal transpositions and, by this, the minimal SWAP gate cascade realizing the desired permutation can be obtained. Using state-of-the-art PBO solvers enables the exploitation of intelligent decision heuristics, powerful learning schemes, and efficient implication methods and, hence, is much more sufficient that simply traversing the complete space of assignments.

C. Optimal SWAP Gate Insertion

Finally, the actual determination of SWAP gates needed in order to make an arbitrary quantum circuit nearest neighbor-compliant is considered. For this purpose, (1) all possible permutations of qubit positions before each gate $g \in G$ of the circuit and (2) the costs (in terms of adjacent SWAP gates) that would be needed in order to create these particular permutations have to be considered. How to calculate the second issue has already been covered in the previous section. For the first issue, again a PBO formulation is proposed.

Again, Boolean variables are introduced for this purpose. We distinguish thereby between the *position* in a circuit and the corresponding *qubits*. Before each gate, we allow an arbitrary permutation (including the identity) which may lead to different mappings of qubits to the respective positions. More precisely:

Definition 4. *Let $G = g_1 g_2 \dots g_{|G|}$ be a quantum circuit over n qubits to be realized in a d-dimensional architecture. Additionally, let each dimension of this d-dimensional quantum circuit be bounded by $b_0, \dots, b_{d-1}$, i.e. the circuit inherits* positions *defined by $P^d \subset \mathbb{N}^d = \mathbb{N} \times \dots \times \mathbb{N}$ with $\forall p = (p_0, \dots, p_{d-1}) \in P^d : p_l \leq b_l$ with $0 \leq l < d$. Then, Boolean variables x_{ij}^k with $1 \leq k \leq |G|$, i enumerating all positions $p_i \in P^d$, and $1 \leq j \leq n$ are introduced representing whether a qubit q_j is assigned a position p_i before gate g_k ($x_{ij}^k = 1$) or not ($x_{ij}^k = 0$)[3].*

Example 3. *Consider again the quantum circuit over five qubits $q_0, \dots q_4$ to be realized in a 3×3 2D architecture as sketched in Fig. 2(b). Additionally assume that the positions $p_i \in P^2$ are enumerated from left to right and top to bottom, i.e. p_0 (p_8) represents the position at the top-left (bottom-right). This permutation to be established before a gate g_k would be represented by the assignment $x_{40}^k = 1$ (qubit q_0 at position p_4), $x_{11}^k = 1$ (qubit q_1 at position p_1), $x_{23}^k = 1$ (qubit q_2 at position p_3), $x_{53}^k = 1$ (qubit q_3 at position p_5), and $x_{74}^k = 1$ (qubit q_4 at position p_7). All remaining x_{ij}^k-variables are assigned zero.*

Obviously, these mappings cannot arbitrarily be made. In fact, each position must exactly correspond to one qubit and each qubit must exactly correspond to one position. In order to ensure this, the following constraint is added to the PBO instance:

$$\bigwedge_{k=1}^{|G|} \bigwedge_{p \in P^d} \left((\sum_{p' \in P^d} x_{p'p}^k = 1) \wedge (\sum_{p' \in P^d} x_{pp'}^k = 1) \right)$$

Next, it has to be ensured that only permutations are applied which satisfy the nearest neighbor condition on all functional gates. Since the control and target qubits of each elementary 2-qubit gate (denoted by $g(c,t)$ with $c, t \in P^d$) are known, this can be enforced through the $\vec{x}_i^k$-variables and the following constraint:

$$\bigwedge_{g_k(c,t) \in G} \bigvee_{i=0}^{d-1} \bigvee_{\substack{p \in P^d | \\ p_i < b_i - 1}} \left((x_{pc}^k \wedge x_{(p+u_i)t}^k) \vee (x_{pt}^k \wedge x_{(p+u_i)c}^k) \right)$$

where u_i is the ith unit vector[4]. More precisely, this constraint enumerates all possible adjacent positions for the

[3] Note that, in accordance to previous work (e.g. [5, 7, 8, 9]), we assume the primary input qubits can arbitrarily be permuted with no additional costs.

[4] Adding the unit vector to a position, i.e. $p + u_i$, means incrementing the ith element of the tupel $p \in P^d$. The result is the next position in the respective direction of the selected dimension and is valid due to $p_i < b_i - 1$.

qubits c and t and eventually ORs them. By this, only assignments are valid which make qubit c and qubit t adjacent.

Finally, the possible permutations of qubits at each position and the corresponding costs for creating such a permutation has to be formulated into the PBO instance. Again, the $\vec{x}_i^k$-variables can be exploited for this purpose. Based on them, it can be derived what permutation is applied before gate g_k in order to change the previous positioning. Further free Boolean variables (denoted by s_π^k) are utilized to store whether a corresponding permutation π is applied. This is expressed by the following constraint:

$$\bigwedge_{k=2}^{|G|} \bigwedge_{\pi \in \Pi} \left(\bigwedge_{p \in P^d} (\vec{x}_p^{k-1} = \vec{x}_{\pi(p)}^k) \Leftrightarrow s_\pi^k \right)$$

This constraint considers all possible permutations (denoted by Π) established before each gate g_k. If the assignments of $\vec{x}_i^{k-1}$ and $\vec{x}_i^k$ establish a particular permutation $\pi \in \Pi$, then the respective variable s_π^k is set to 1 (encoded through $\Leftrightarrow$). This states that this particular permutation π has been chosen before gate g_k and, hence, the corresponding costs for it have to be considered. This is eventually incorporated in the objective function

$$min : \sum_{k=2}^{|G|} \sum_{\pi \in \Pi} c_\pi s_\pi^k$$

where c_π denotes the costs (in terms of adjacent SWAP gates) for creating a permutation π. These costs have been determined before as described in Section IV.B.

Combining all these constraints, a PBO instance results which is satisfiabile for all permutations of qubits that lead to a nearest neighbor compliant circuit. The precise permutation to be created at position k can thereby be derived from the assignment to the s_π^k variables. If s_π^k has been assigned 1 by the PBO solver, a permutation π has to be created before gate g_k. By additionally optimizing the objective function, the PBO solver ensures a minimal number of SWAP gates.

V. Application and Discussion

The approaches presented above address, in an exact fashion, the main issues to be solved when it comes to determine the minimal number of SWAP gates in order to make a multi-dimensional quantum circuit nearest neighbor-compliant. In this section, we show and discuss how this advances the state-of-the-art in this domain. Due to page limitations, we limit ourselves to selected examples which representatively illustrate the benefits but also the limitations of the proposed approaches.

A. Costs of Establishing an Arbitrary Permutation

SWAP gate insertion basically is about establishing a new permutation of qubits to the respective positions within a multi-dimensional quantum circuit. Hence, before it comes to an actual SWAP gate insertion, the general question is how costly (in terms of SWAP gates) is it to realize a certain permutation. While linear solutions to this question exist for 1D circuits (due to the help of inversion vectors [8, 9]), the approach presented in Section IV.B represents the first exact solution for multi-dimensional circuits. This does not only provide the designer with crucial information on the (exact) costs of establishing a certain operation, but also allows an analysis on the suitability of different configurations with respect to nearest neighbor constraints.

TABLE I
Costs of establishing an arbitrary permutation

π	#SWAPs 1D	#SWAPs 2D	π	#SWAPs 1D	#SWAPs 2D
$q_0q_1q_2q_3$	0	0	$q_2q_0q_1q_3$	2	2
$q_0q_1q_3q_2$	1	1	$q_2q_0q_3q_1$	3	3
$q_0q_2q1q_3$	1	3	$q_2q_1q_0q_3$	3	1
$q_0q_2q_3q_1$	2	2	$q_2q_1q_3q_0$	4	2
$q_0q_3q_1q_2$	2	2	$q_2q_3q_0q_1$	4	2
$q_0q_3q_2q_1$	3	1	$q_2q_3q_1q_0$	5	3
$q_1q_0q_2q_3$	1	1	$q_3q_0q_1q_2$	3	3
$q_1q_0q_3q_2$	2	2	$q_3q_0q_2q_1$	4	2
$q_1q_2q_0q_3$	2	2	$q_3q_1q_0q_2$	4	2
$q_1q_2q_3q_0$	3	3	$q_3q_1q_2q_0$	5	3
$q_1q_3q_0q_2$	3	3	$q_3q_2q_0q_1$	5	3
$q_1q_3q_2q_0$	4	2	$q_3q_2q_1q_0$	6	4

As a representative, Table I shows the obtained costs needed to establish all $4! = 24$ possible permutations over 4 qubits in both, a 1D quantum circuit as well as a 2×2, i.e. 2D, quantum circuit. The first column denotes thereby all possible permutations π, while the remaining two columns provide the number of SWAP gates needed in order to realize the respective π's in the 1D circuit and the 2D circuit.

Obviously, establishing the identify permutation (i.e. $q_0q_1q_2q_3$) does not require a SWAP gate in neither the 1D nor the 2D configuration. But for all other permutations, significant differences can be observed. In fact, 2D architectures require never more than 4 SWAP gates – in a single case only. Instead, 1D circuits require 4 SWAP gates or more (i.e. up to 6) in a total of 9 cases. Hence, with respect to nearest neighbor constraints, the higher dimension certainly pays off. But this does not necessarily hold for all permutations. For example, the permutation $q_0q_2q_1q_3$ can be realized in a 1D architecture with a single SWAP gate only, why 3 SWAP gates are needed in a 2D circuit. Without a scheme which enables logic designers to determine those exact values, precise conclusions as discussed here would not be possible.

B. Optimal SWAP Gate Insertion

Considering the actual SWAP gate insertion for a particularly given circuit, very few results exist yet. All of them are of heuristic nature. Here, the approach proposed in Section IV.C advances the state-of-the-art by, for the first time, providing exact solutions. This allows for an evaluation on how far the previously obtained heuristic results are from the optimum.

Table II provides a selection of results confirming this statement. Here, the heuristically determined number of SWAP gates as reported in [17] is compared against the exactly determined number of SWAP gates obtained by the approach proposed in Section IV.C (the columns *Circuit* and *Conf.* provide the name and the configuration as used in [17], respectively).

TABLE II
RESULTING OPTIMAL SWAP GATE INSERTION

Circuit	Conf.	#SWAPs [17]	#SWAPs Sect. IV.C	Time
3_17_13	2x2	6	4	17.6s
decod24-v3_46	2x2	3	2	0.5s
hwb4_52	2x2	9	7	42752.9s
rd32-v0_67	2x2	2	2	0.2s
4gt11_84	2x3	1	1	1644.5s

As expected, the approach presented in [17] – to the best of our knowledge the only solution addressing SWAP gate determination for 2D architectures thus far – does not guarantee minimality. In fact, (exact) solutions with less SWAP gates are possible as shown by means of the first three representatives in Table II. On the other side, this does not mean that the approach from [17] never realizes minimal solutions. In fact, as shown in the last two rows of Table II, respective representatives exists. However, without the approach presented in Section IV.C, it would still be unknown whether these results are indeed minimal or not.

In all these evaluations, the computation time remains the limiting factor. That was expected and is a well-known characteristic of exact synthesis schemes in general (regardless of whether conventional or emerging technologies are considered). In our evaluations, we were able to determine exact results for configurations with up to 6 qubits (using an AMD Athlon X2 CPU with 3 GHz and 4 GB of memory). This is in accordance to exact synthesis schemes for other purposes. The right-most column of Table II gives the detailed run-times for the selected representatives. Despite this limitation, the proposed approaches nevertheless advances the field of nearest neighbor optimization for multi-dimensional quantum circuits by providing the minimal number of SWAP gates needed in order to establish an arbitrary permutation for several circuit configurations and a methodology to realize minimal solutions to be used for comparison to (much more scalable) heuristic results.

VI. CONCLUSIONS

In this work, we considered the problem of how to exactly determine the minimal number of SWAP gates to be inserted in order to make a generic, i.e. multi-dimensional, quantum circuit nearest neighbor-compliant. We observed that a solution for this problem requires several steps – each of them with its certain complexity. To cope with the respective complexity, solvers for pseudo-Boolean satisfiability have been utilized. This allowed, for the first time, for a qualitative evaluation of the respective optimization steps and enabled an exact comparison to heuristical results.

REFERENCES

[1] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[2] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Foundations of Computer Science*, pages 124–134, 1994.

[3] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Theory of computing*, pages 212–219, 1996.

[4] Y. Hirata, M. Nakanishi, S. Yamashita, and Y. Nakashima. An efficient method to convert arbitrary quantum circuits to ones on a Linear Nearest Neighbor architecture. *Quantum, Nano and Micro Technologies. ICQNM*, pages 26–33, 2009.

[5] M. Saeedi, R. Wille, and R. Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quant. Info. Proc.*, 10(3):355–377, 2011.

[6] Atsushi Matsuo and Shigeru Yamashita. Changing the gate order for optimal LNN conversion. In *Reversible Computation*, volume 7165 of *Lecture Notes in Computer Science*, pages 89–101. Springer Berlin Heidelberg, 2012.

[7] A. Shafaei, M. Saeedi, and M. Pedram. Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In *Design Automation Conf.*, pages 41–46, 2013.

[8] Robert Wille, Aaron Lye, and Rolf Drechsler. Optimal SWAP gate insertion for nearest neighbor quantum circuits. In *ASP Design Automation Conf.*, pages 489–494, 2014.

[9] R. Wille, A. Lye, and R. Drechsler. Exact reordering of circuit lines for nearest neighbor quantum architectures. *IEEE Trans. on CAD*, 33(12), 2014.

[10] L. C. L. Hollenberg, A. D. Greentree, A. G. Fowler, and C. J. Wellard. Two-dimensional architectures for donor-based quantum computing. *Phys. Rev. B*, 74:045311, 2006.

[11] Muir Kumph, Michael Brownnutt, and Rainer Blatt. Two-dimensional arrays of radio-frequency ion traps with addressable interactions. *New Journal of Physics*, 13(7):073043, 2011.

[12] Naomi H. Nickerson, Ying Li, and Simon C. Benjamin. Topological quantum computing with a very noisy network and local error rates approaching one percent. *Nat Commun*, 4:1756, 2013.

[13] Alexandre Blais, Jay Gambetta, A. Wallraff, D. I. Schuster, S. M. Girvin, M. H. Devoret, and R. J. Schoelkopf. Quantum-information processing with circuit quantum electrodynamics. *Phys. Rev. A*, 75:032329, Mar 2007.

[14] J. M. Taylor, J. R. Petta, A. C. Johnson, A. Yacoby, C. M. Marcus, and M. D. Lukin. Relaxation, dephasing, and quantum control of electron spins in double quantum dots. *Phys. Rev. B*, 76:035315, Jul 2007.

[15] M. Saffman, T. G. Walker, and K. Mølmer. Quantum information with Rydberg atoms. *Rev. Mod. Phys.*, 82:2313–2363, Aug 2010.

[16] B.-S. Choi and R. Van Meter. A $\theta(\sqrt{n})$-depth quantum adder on the 2D NTC quantum computer architecture. *J. Emerg. Technol. Comput. Syst.*, 8(3):24, 2012.

[17] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. Qubit placement to minimize the communication overhead in circuits mapped to 2D quantum architectures. In *ASP Design Automation Conf.*, pages 495–500, 2014.

[18] N. Y. Yao, Z.-X. Gong, C. R. Laumann, S. D. Bennett, L.-M. Duan, M. D. Lukin, L. Jiang, and A. V. Gorshkov. Quantum logic between remote quantum registers. *Phys. Rev. A*, 87:022306, 2013.

[19] David A. Herrera-Martí, Austin G. Fowler, David Jennings, and Terry Rudolph. Photonic implementation for the topological cluster-state quantum computer. *Phys. Rev. A*, 82:032332, 2010.

[20] R. Wille, O. Keszöcze, and R. Drechsler. Determining the minimal number of lines for large reversible circuits. In *Design, Automation and Test in Europe*, pages 1204–1207, 2011.

[21] D. M. Miller, R. Wille, and R. Drechsler. Reducing reversible circuit cost by adding lines. In *Int'l Symp. on Multi-Valued Logic*, pages 217–222, 2010.

[22] R. Wille, M. Soeken, D. M. Miller, and R. Drechsler. Trading off circuit lines and gate costs in the synthesis of reversible logic. *INTEGRATION, the VLSI Jour.*, 47(2):284–294, 2014.

[23] F.A. Aloul and B.A. Rawi. Identifying the shortest path in large networks using Boolean satisfiability. In *Electrical and Electronics Engineering*, pages 1–4, Sept 2006.

[24] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. *clasp*: A conflict-driven answer set solver. In *Logic Programming and Nonmonotonic Reasoning*, pages 260–265, 2007.