Graph-Theoretical Arguments in Support of a Quantum Declarative Manifesto

Alex Della Schiava[1], Carla Piazza[1] and Riccardo Romanello[1]

[1]*Dipartimento di Scienze Matematiche, Informatiche e Fisiche, Università degli Studi di Udine*

Abstract

The encoding of graphs and the development of efficient algorithms over graphs in the Quantum framework is still a challenging problem. More in general, the understanding of whether a problem can benefit of the Quantum speed-up or not is far from being complete. In this paper we analyse, compare, and generalize some proposals for the encoding of graphs in Quantum computing. A question peeping out on the horizon of our analysis concern the shift from a procedural way of thinking to a declarative one in the context of Quantum computing.

Keywords

Quantum computing, Graphs, Quantum walks, Quantum speed-up

Introduction

In May 1981, Feynman gave a conference lecture on "Simulating physics with computers". In that occasion, he proposed the idea of using quantum computers to simulate quantum systems that are too hard to simulate using classical computers. His talk, published in [1], is remembered as the big bang of quantum computing as a research field.

Since that moment, both academia and companies have put a lot of effort in the task of developing useful Quantum Computers. Despite the stakeholders that joined the task, the path is far from being complete. One of the biggest problems is a manufacturing one: creating and preserving qubits. Qubits are the equivalent of bits in the quantum setting, but they are not as easy to preserve as their classical counterpart. They must be stored at a very low temperature in order to maintain coherence — any computation based on uncoherent qubits leads to error. Despite the high costs demanded by the task, a handful of companies have been able to construct working Quantum Computers.

Confined by the rules of quantum mechanics, the quantum model of computation solely manipulates qubits through unitary operations, which are a subset of linear operators. Projections, another subset of linear operators, allow to read the content of qubits, albeit by inevitably altering their state. Hence, every Quantum algorithm must be developed as a sequence of unitary operators and projections. Quantum algorithms such as Shor's factorization [2] and

CILC'23: 38th Italian Conference on Computational Logic, June 21–23, 2023, Udine, Italy

[†]This work is partially supported by PRIN project NiRvAna and GNCS-INdAM project "Algoritmi Quantistici su Grafi" CUP_E53C22001930001.

dellaschiava.alex@spes.uniud.it (A. Della Schiava); carla.piazza@uniud.it (C. Piazza); riccardo.romanello@uniud.it (R. Romanello)

Grover's search [3] witness the different kind of speed-up achievable by adopting a quantum system as model of calculus. Nevertheless, there exists no standard recipe to systematically exploit the properties of quantum mechanics to obtain similarly remarkable results. For example, problems defined over graphs play a central role in Computer Science. Since their introduction by Euler in [4]–which is considered the first paper in Graph Theory–graphs have become more and more prominent in mathematics first and in computer science later. Not only do they allow to model and solve a vast amount of problems, but they also capture the notion of computation over procedural models such as Turing Machines, Random Access Machines, and imperative programming languages [5].

As a consequence, to understand and generalize quantum speed-up results, one could investigate how graphs and graph algorithms behave on quantum architectures. Nevertheless, graphs did not receive much attention in the quantum case: there are no exhaustive answers as to how these data structures may be encoded in the quantum setting. Current techniques require graphs to satisfy severely restrictive properties in order to be encoded [6, 7]. Starting from assumptions of this form, quantum algorithms over graphs are not general. Moreover, graphs have hardly been adopted and used in theoretical Quantum Computing to show results about expressiveness and complexity. It is enough to dig a little into the topic of Quantum Automata to see how many questions are still open [8].

In light of these theoretical questions, the use of Quantum Hardware is hardly justifiable: only a handful of algorithms fully exploit its benefits. While efforts have been directed toward defining further efficient quantum algorithms through Quantum Programming Languages [9, 10], in this paper we prompt investigations on the efficient compilation of declarative languages over quantum architectures. Joint efforts on the investigations would reciprocally benefit the two involved research fields: How can declarative programming and Quantum Computing assist each other?

Our intuition starts by observing the difficulties arising in the quantum encoding and management of graphs. These complications are strongly related to the intrinsic reversibility of quantum computation. Intuitively, only strongly connected graphs are reversible. As a consequence, an efficient compilation of procedural languages that rely on operational semantics, i.e., on graphs, looks difficult. On the other hand, because of their "natural" denotational semantics, declarative languages seem like better candidates. While the main part of this paper is devoted to give the reader evidence on the obstacles in using graphs within quantum computation, in Section 4 we come back to our claim on declarative languages.

The paper is structured as follows. In Section 1 we give an overview of Quantum Computing from two different perspectives. The former builds up from a lower level, introducing the elements from linear algebra required to handle theoretical Quantum Computing. The latter is a high level view where we lay down the mathematical details to focus on the general ideas of quantum computation. The purpose of Section 2 is to introduce and explain the key role graphs play in classical computation. Section 3 thus shifts the discussion to the quantum setting, providing insights on the problem of encoding graphs into quantum computation. In Section 4 we speculate on the possible advantages of a shift of point of view in the direction of compilers from classical declarative languages on quantum machines.

1. From Low to High Level View of Quantum Computation

1.1. Low Level View

The most used model of Quantum Computation relies on the formalism of state vectors, unitary operators and projectors. State vectors evolve during the computation through unitary operators, then projectors are used to remove part of the uncertainty on the internal state of the system.

The state of the system is represented by a unitary vector over $\mathbb{C}^n$ with $n = 2^m$ for some $m \in \mathbb{N}$. The concept of a bit of classical computation is replaced by that of a qubit. While bits have value either 0 or 1, qubits are unitary vectors in $\mathbb{C}^2$. The two components of the qubit are the complex numbers $\alpha = x + iy$ and $\beta = z + iw$; their squared norms $|\alpha|^2 = x^2 + y^2$ and $|\beta|^2 = z^2 + w^2$ represent, respectively, the probability of measuring the qubit at either value 0 or 1. In the more general case of m qubits the unitary vectors range in $\mathbb{C}^n$ with $n = 2^m$. Adopting the standard Dirac notation we denote a column vector $v \in \mathbb{C}^n$ by $|v\rangle$, and its conjugate transpose $v^\dagger$ by $\langle v|$. A *quantum state* is a unitary vector

$$|\psi\rangle = \sum_h c_h |v_h\rangle$$

for some basis $\{|v_h\rangle\}$. When not specified, we refer to the *canonical basis.* Further details can be found in [11].

Unitary operators and projectors are linear operators. *Unitary operators* are a particular class of reversible linear operators. They preserve both the angles between vectors and their lengths. In other terms, unitary operators are transformations from one orthonormal basis to another. Hence, they are represented by unitary matrices. Let U be a square matrix over $\mathbb{C}$. U is said to be *unitary* iff $UU^\dagger = U^\dagger U = I$. We describe the application of a unitary matrix U to a state $|\psi\rangle$ by writing

$$|\psi'\rangle = U |\psi\rangle$$

meaning that the state $|\psi\rangle$ becomes $|\psi'\rangle$ after applying the operator U.

In order to extract information from a quantum state $|\phi\rangle$ a *measurement* must be performed. The most common measurements are projectors. Let $|u\rangle$ be a quantum state. The *projector* operator P_u along the direction of $|u\rangle$ is the linear operator defined as:

$$P_u = \frac{|u\rangle\langle u|}{\langle u|u\rangle}$$

where $|u\rangle\langle u|$, being the product between a column vector and a row one both of size n, returns a matrix of size $n \times n$. Since throughout the paper we only use unitary vectors, the term $\langle u|u\rangle$ is always 1 and can be ignored.

1.2. High Level View

In order to both summarize and take a step back from technical details, we can say that quantum computation is a computation over vectors of complex numbers which only exploits a subset of linear functions (i.e., unitaries and projectors) as basic operations. Each function that can be obtained as a composition of these basic operations is a quantum circuit. The model becomes

Turing complete by either considering uniform families of circuits or by adding loops over classical Boolean variables (e.g., Boolean values obtained through projectors).

One of the main peculiarities of quantum computation is that of being *reversible*. As a matter of fact once a unitary operator U has modified the system, we can go back to the past state by applying the unitary operator $U^\dagger$.

Classical computation is in general non-reversible. Let us consider the conjunction Boolean operator $\wedge$. When we apply it to the bits x and y, we obtain the output bit z. From z it is not always possible to infer the values of x and y. However, any classical Boolean function f

$$f : \{0,1\}^m \to \{0,1\}$$

can be embedded into the reversible function $\overline{f}$

$$\overline{f} : \{0,1\}^{m+1} \to \{0,1\}^{m+1}$$

defined as

$$\overline{f}(x,y) = (x, y \oplus f(x))$$

where $x \in \{0,1\}^m$ and $y \in \{0,1\}$. Not only $\overline{f}$ is reversible, but it is unitary and it coincides with its inverse.

If we simply think at quantum computation as a sequence of unitary and projector operators applied to an initial vector, it shares some aspects with the computation done within a neural network. Both unitaries and projectors are linear transformations. So, Quantum compared with Neural Networks lacks of the key feature of non-linear activation functions. However, quantum computation is performed over complex numbers, while a neural network relies on the reals. As witnessed by quantum interference, a complex number is much more than a pair of reals.

The main phenomena that distinguish quantum computation from the classical one are: superposition, interference and entanglement.

The quantum state $|\psi\rangle = \sum_h c_h |v_h\rangle$ is in a *superposition* of classical states and a projector is necessary to collapse it in the classical world.

Interference allows probabilities to behave in an unexpected way. In the quantum framework two events A and B can have both positive probabilities, while $A \vee B$ has probability 0.

Entanglement is so surprisingly and unexplainable that even Einstein could not believe it. It is a sort of correlation between particles that remains true until they are observed.

2. Graphs at the Basis of Procedural Computation Models

Graphs are a standard data structure in Computer Science for the representation of binary relations. We report below some standard definitions on graphs, while we refer the reader to [12] for further details. A directed *graph* G is a pair (V, E) where V is a non empty set of *vertices* and $E \subseteq V \times V$ is a set of *edges*. We say that an edge (u, v) is *directed* from u to v and that u is *adjacent* to v. We also say that u is the *source* and v is the *target* of (u, v). Two edges of the form (u, v) and (v, w) are said to be *consecutive*. Given a vertex v, its *out-neighbourhood* $\delta^+(v)$ (*in-neighbourhood* $\delta^-(v)$) is the set of vertices connected via an edge *from* (*to*) v; the *out-degree* (*in-degree*) of v is $d^+(v) = |\delta^+(v)|$ ($d^-(v) = |\delta^-(v)|$).

Graphs are the key data structure for efficiently solving problems defined over maps, networks, processes and, more in general problems, that can be stated through binary relations. Not only graphs allow to model and efficiently solve such a wide range of problems, but they are at the core of every computation endowed with an operational semantics. As a consequence, time and space complexities of the *reachability problem* over deterministic and non-deterministic Turing machines are at the basis of the most important results in Complexity Theory, such as Savitch's theorem and Immerman-Szelepscényi theorem [5].

In the context of randomized computations, again graphs, and in particular *random walks* over graphs, play a central role both in the design of efficient algorithms and in the development of general strategies for speeding-up their convergence [13].

So, on the one hand, one could observe that any computational model that aims at solving interesting problems should provide a set of tools for managing graphs (i.e., for solving problems over graphs). On the other hand, a computational model able to efficiently solve the reachability problem has already the ability to *efficiently* simulate classical machines, and can aim to beat them over specific problems.

Such point of view naturally suggests some questions in the direction of a better understanding of the potential of quantum computation:

- Which problems over graphs can be efficiently solved over a quantum model?
- How can graphs be efficiently stored in a quantum machine?

At the moment the answers to the above questions are not complete. Several approaches have been presented in the literature and we will provide details on some of them in the next sections. With different strategies these approaches try to encode graphs into unitary matrices. Intuitively, during a traversal/walk over the graph the state vector keeps track of the current superposition of vertices, which looks amazing, since it means that we are "in parallel" proceeding along different directions. Differently from classical random walks, some paths may interfere destructively with each other, introducing bias in the global behaviour of the quantum walk. Remarkably, this action significantly reduces the time required by the quantum walk to spread across certain graphs [14, 15]. Moreover, only some classes of graphs ensure the reversibility of the traversal, while all the others have to be somehow embedded in larger structures.

Are these unsatisfactory answers to the above questions just a matter of how much effort has been put so far on the topic? Cannot be the case that they are a symptom of the need for a shift of point of view? We will come back to this in the last section.

3. Graph Encoding in Quantum

In classical computations graphs are largely used as an abstract data structure where to encode the input knowledge and efficient graph algorithms are developed with the aim of computing the desired output.

On the contrary, in the context of Quantum Computation graphs are mainly studied as the domain for quantum walks. As a matter of fact, quantum walks have been a fundamental tool in physics, where they have been employed to analyze/simulate the evolution of quantum systems.

Quantum walks are the quantum mechanical counterpart of random walks. A *“quantum walker"* moves analogously to a classical one, albeit, with three key differences: (i) he may lie in a *superposition* of states; (ii) his steps are guided by *probability amplitudes*; (iii) only unitary transformations can be used to guide the changes of the quantum walker. In the context of quantum simulation, the quantum walker is meant to represent the quantum system under consideration and, as such, he obeys the laws of Quantum Mechanics. Indeed, conditions (i), (ii), and (iii) given above are but a coarse summary of these laws. Conditions (i) and (ii)—together with entanglement—are at the basis of quantum speed-up. Condition (iii) poses a strong limit by restricting the class of graphs over which walks may be performed. Nonetheless, both continuous and discrete-time quantum walks have been shown to be as powerful as the quantum circuit model of computation [16, 17]. However, issues arise when studying quantum walks and quantum representations of graphs in a more general setting.

For the above reasons in this section we investigate the classes of graphs amenable to representations in the quantum setting and to quantum walks. Then, we analyse how general graphs can be embedded into ones belonging these classes as well as the costs such embedding procedures demand.

3.1. Directed Graphs and Unitary Matrices

The relationship between graphs and unitary matrices is formalized via the adjacency matrix representation. As a preliminary notion, the *support* of a matrix $M \in \mathbb{C}^{n\times n}$ is defined as the matrix $M^S \in \{0,1\}^{n\times n}$ such that, for any $1 \leq i, j \leq n$,

$$M^S_{ij} = \begin{cases} 1 & \text{if } M_{ij} \neq 0; \\ 0 & \text{otherwise.} \end{cases}$$

A graph and a unitary matrix may, thus, be related in terms of the following definition.

Definition 1. A graph G is said to be the *graph of a unitary matrix* U if and only if the respective adjacency matrix $M(G)$ supports U.

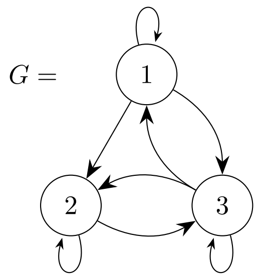
Example 1. Graph G depicted in Figure 1 is the graph of the unitary matrix U,

$$U = \begin{pmatrix} 1/\sqrt{2} & 1/2 & 1/2 \\ 0 & 1/\sqrt{2} & -1/\sqrt{2} \\ -1/\sqrt{2} & 1/2 & 1/2 \end{pmatrix}.$$

Graphs of unitary matrices are amenable to quantum walks. In the case of graph G from Figure 1, a quantum walk may be defined in terms of operator U. Its construction is analogous to that of a classical Markov chain. The three vertices of G induce the state space $\mathbb{C}^3$, with each vertex v being represented by a column vector $|v\rangle$ of the canonical basis. Transitions are guided by the adjoint of U: $U^\dagger$. The adjoint $U^\dagger$ is here required to abide to the conventional representation of unitary evolution as a matrix-vector multiplication.

To provide an example, let the quantum walk start from state $|1\rangle = (1,0,0)^T$. Performing the first step then leads to state

$$U^\dagger |1\rangle = \frac{1}{\sqrt{2}} |1\rangle + \frac{1}{2} |2\rangle + \frac{1}{2} |3\rangle .$$



$$M(G) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Figure 1: Graph of a unitary matrix.

After a single step, the quantum walker finds herself into a superposition of all three vertices of the graph.

Another insightful example sees the quantum walk start from state $|\psi\rangle = \frac{1}{2}|1\rangle + \frac{1}{\sqrt{2}}|2\rangle + \frac{1}{2}|3\rangle$. A single step transition leads to state

$$U^\dagger |\psi\rangle = |2\rangle .$$

The step has produced a rather singular effect: whereas the probability amplitudes for the paths leading to vertex 2 reinforce each other, the paths heading towards vertices 1 and 3 cancel out. These two phenomena are known, respectively, as *constructive* and *destructive interference* and characterize the peculiar behaviour of quantum walks: *more paths heading towards the same vertex do not necessarily imply a higher probability of reaching it.*

As previously mentioned, the class of graphs of unitary matrices is severely restrictive. Moreover, it is hard to characterize the class through ordinary properties from Graph Theory. From the literature, three distinct approaches to the problem emerge, each relating graphs of unitary matrices to other graph properties: (i) via *strong quadrangularity* [18]; (ii) via *graph reversibility* [6]; (iii) via study of *bridges* and *cuts* [19]. As for (iii), the work explores how different connected components from graphs of unitary matrices are connected to each other. The study is here not elaborated any further.

Strong Quadrangularity In [18], graphs of unitary matrices have been studied with respect to the property of *strong quadrangularity.*

Definition 2 (Strong Quadrangularity)**.** Given a graph $G = (V, E)$, consider sets of the two following forms:

- $S_{\text{out}} \subseteq V$, where, for any $u \in S_{\text{out}}$ there exists $v \in S_{\text{out}}$ such that $\delta^+(u) \cap \delta^+(v) \neq \emptyset$.
- $S_{\text{in}} \subseteq V$, where, for any $u \in S_{\text{in}}$ there exists $v \in S_{\text{in}}$ such that $\delta^-(u) \cap \delta^-(v) \neq \emptyset$.

Then, G is said to be *strongly quadrangular* if and only if for any set of the form S_{out} or S_{in} it holds that

$$\left| \bigcup_{u,v \in S_{\text{out}}} \delta^+(u) \cap \delta^+(v) \right| \geq |S_{\text{out}}|; \qquad \left| \bigcup_{u,v \in S_{\text{in}}} \delta^-(u) \cap \delta^-(v) \right| \geq |S_{\text{in}}|.$$

Although apparently abstract, the property of strong quadrangularity provides deep, matrix-theoretical insights over the class of graphs of unitary matrices.

Lemma 1. *[18] Any graph of a unitary matrix is strongly quadrangular.*

The opposite direction of this relationship involves the property of *specularity.*

Definition 3 (Specularity)**.** A graph $G = (V, E)$ is said to be *specular* if and only if, for any pair of vertices $u, v \in V$, the following two conditions are satisfied:

$$\delta^+(u) \cap \delta^+(v) = \emptyset \text{ or } \delta^+(u) = \delta^+(v); \quad \text{and} \quad \delta^-(u) \cap \delta^-(v) = \emptyset \text{ or } \delta^-(u) = \delta^-(v).$$

In other words, two vertices of a specular graph either share the entire in/out-neighbourhood or none of it. Pairing specularity and strong quadrangularity gives the following result.

Theorem 1. *[18] A specular, strongly quadrangular graph is the graph of a unitary matrix.*

A broader version of the discussion involves the notion of *line graph.*

Definition 4 (Line Graph)**.** Given a graph $G = (V, E)$, the respective line graph is $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$ where:

- $\overrightarrow{V} = E$. The vertices in $\overrightarrow{G}$ represent the edges from G.
- $\overrightarrow{E} = \{\big((u, v), (v, w)\big) : (u, v), (v, w) \in E\}$. Two vertices are adjacent in $\overrightarrow{G}$ if and only if they represent consecutive edges in G.

The convenience in dealing with line graphs is twofold. On the one hand, a line graph encodes all adjacencies of the original graphs. On the other, line graphs are tightly linked to the property of specularity, as stated by the following result.

Lemma 2. *Any line graph is specular.*

Theorem 1 now prompts the question as to which graphs induce strongly quadrangular line graphs. As it turns out, the question is answered by Eulerian graphs or graphs that are disjoint unions of Eulerian components.

Theorem 2. *[18] Let G be a graph. Then, $\overrightarrow{G}$ is the graph of a unitary if and only if G is Eulerian or the disjoint union of Eulerian components.*

As for Theorem 2, it should be clarified that the proof from left to right is only partially related to the reasoning constructed so far in this section.

Graph Reversibility Following a different path, Montanaro studied graphs of unitary matrices from the perspective of *graph reversibility* [6]. This novel notion builds upon its local version: an edge (u, v) is *reversible* if and only if there exists a path from v to u.

Definition 5 (Graph Reversibility)**.** [6] A graph is said to be *reversible* if and only if all of its edges are reversible.

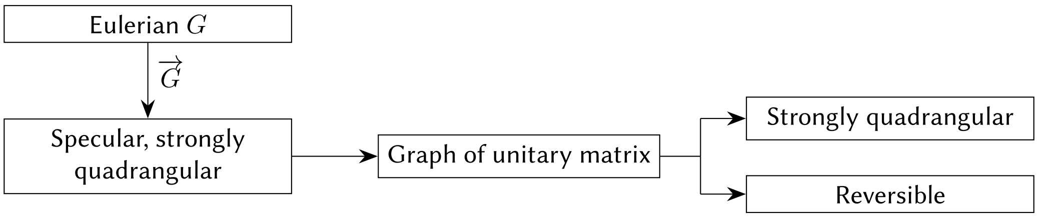


Figure 2: Graphs of unitary matrices in Graph Theory.

Reversibility is closely tied with the more standard property of strong connectivity: any strongly connected graph is reversible. However, the opposite is not true: the class of reversible graphs includes all graphs composed of different disconnected strongly connected components.

The following result provides the main connection between graphs of unitary matrices and reversibility.

Theorem 3. *[6] If G is the graph of a unitary matrix then G is reversible.*

In light of these results, strong quadrangularity and reversibility appear to relate to graphs of unitary matrices in distinct ways. Whereas the study of strong quadrangularity requires a deeper quasi matrix-theoretical analysis, reversibility links to graphs of unitary matrices on a higher level. Because quantum computation is inherently reversible, it is of no surprise that graphs apt to describe it should satisfy some sort of Graph Theoretical-analogous of reversibility.

To clarify the picture described by the results provided so far, Figure 2 provides a conceptual summary of the relationships between graphs of unitary matrices and the graph properties here reviewed.

3.2. Encoding Graphs into Unitary Matrices

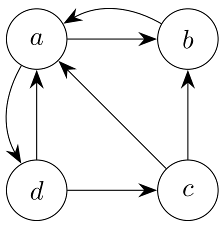
Section 3.1 provided insights over the properties that characterize graphs of unitary matrices. Because these are the graphs that lend themselves to quantum walks, a reasonable question would ask how to transform general graphs into graphs of unitary matrices. *"Graph Encoding in Quantum Computing"* refers precisely to this transformation.

Hereby, two *edge addition based* encoding procedures are summarized: from general to Eulerian graphs [20]; from irreversible to reversible graphs [6].

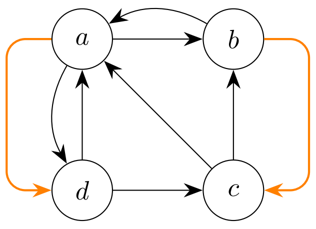
Encoding General Graphs into Eulerian Ones In [20], an algorithmic solution aimed at encoding graphs into unitary matrices was proposed. Because the result solely relies on the assumption that the given multigraph be connected, the procedure may be extended to general graphs via reiteration over each connected component.

The procedure builds upon Theorem 2 through the following steps: (i) the given graph $G = (V, E)$ is encoded into a Eulerian graph G'; (ii) the line graph $\overrightarrow{G'}$ is computed; (iii) G is encoded into a unitary matrix U supported by the adjacency matrix $M(\overrightarrow{G'})$. By Theorem 2, $\overrightarrow{G'}$ is the graph of a unitary matrix, thus U exists.

Providing a closer inspection over the workings of step (i), this exploits the following result.



(a) Input graph G.



(b) Eulerian multigraph G'.

Figure 3: Encoding graph G into the eulerian multigraph G' via edge addition.

Theorem 4. *A connected graph $G = (V, E)$ is Eulerian iff, for any $v \in V$, $d^+(v) = d^-(v)$.*

On these grounds, the procedure *"Eulerifies"* G by balancing the in- and out-degree of each vertex through addition of edges. Figure 3 provides an example of this process, the added edges are highlighted in Figure 3b in orange color. Because edges may be duplicated, G is, more generally, encoded into a Eulerian *multigraph*, that is, a graph where two vertices may be connected by more than one edge going the same direction.

In light of the fact that, in a directed graph, incoming and outgoing edges always come in the same quantity, the procedure can be shown to converge in $\mathcal{O}(|V| + |E|)$ time. All three steps considered, the overall time-complexity of the procedure is $\Theta(|E|^2)$.

As is the case for the example in Figure 3, the encoded graph might include adjacencies that were not part of the original graph. While the encoding itself does not address the issue, projectors do come in to rescue. Let us briefly recall that the state space of a quantum walk is spanned by the vertices of the given graph. Now, because this encoding procedure constructs a quantum walk on a line graph, unwanted edges become unwanted vertices and are thus quickly taken care of with projectors. Elaborating on this, let $E_\perp$ be the set of newly created edges and consider projector

$$P_G = I - \sum_{e \in E_\perp} |e\rangle \langle e| ,$$

where I is the identity operator. Applying P_G on any state of the walk erases any part of the superposition concerning the unwanted edges, thus continuing the walk as they were never traversed. Clearly, P_G needs to be applied after each step of the walk.

Encoding Irreversible Graphs into Reversible Ones Apart from providing a reasonable notion of graph reversibility, Montanaro has also developed a procedure to edit irreversible graphs so to satisfy this newly defined graph property [6]. Before proceeding any further, it should be clarified what reversible graphs are good for; the results given thus far are silent as to the conditions these graphs *must* satisfy. As it turns out, once a reversible graph is equipped with self-loops at each vertex, a procedure exists to encode it into the graph of a unitary matrix. The process works through a specific cycle decomposition of the graph. Further details may be found in [6].

Returning to irreversible graphs, the path towards their transformation to reversible ones begins with the identification of all those edges that are irreversible. Trivially, these edges are to be turned into reversible ones, though defining the resulting quantum walk shall require a more elaborate plan.

Given a graph G, removing all such edges inevitably leads to a reversible graph G^{rev}. Indeed, no reversible edge becomes irreversible through such removal. The effect of the removal is twofold: (i) G^{rev} is structured into different reversible connected (or, equivalently, strongly connected) components $C_1, C_2, \ldots, C_D$; (ii) in G, irreversible edges always connect two distinct components C_i, C_j. Let (u, v) be the irreversible edge connecting components C_i, C_j. C_i is augmented with vertex v. To preserve reversibility of C_i, edge (v, u) is added. Finally two distinct quantum walks W_i, W_j are constructed over reversible components C_i and C_j via the above procedure. Depending on which component the walker is currently visiting, the respecting quantum walk operator is applied.

However, with a solution comes a problem, namely that the quantum walker should be prevented from ever traversing edge (v, u), as it does not belong the original graph G. To this end, projective measurements are devised, such that, upon superposition between vertices of C_i, C_j, the walker collapses to one of the components. That is, for any connected component of G^{rev}, the following projector is defined

$$P_k = \sum_{v \in C_k} |v\rangle \langle v| \,.$$

After a step, the walker is measured via all projectors P_k. Measuring i or j means, respectively, finding the walker in either reversible component C_i or C_j. Thus, given measurement outcome k, the quantum walk is continued via operator W_k. Naturally, the next step could again traverse an irreversible edge. Thus, just as in the encoding procedure into Eulerian graphs, quantum walk operators and projectors need be alternated.

How Truthful Can an Encoding Procedure Be? Both encoding procedures here reviewed appear to heavily rely on the use of projectors to ensure *truthfulness* with respect to the structure of the original graph. More specifically, projectors are inevitable whenever dealing with non-strongly connected graphs: yet another hint to the restriction posed by computational reversibility and graph reversibility.

In this regard, the study on truthful encoding procedures should also concern *duplicated* edges. Indeed, edge duplication also affects graph topology, albeit in a somewhat softer way. In turn, these effects appear to produce phenomena of local bias in the resulting quantum walk.

4. Conclusions and Future Declarative Directions

A large part of quantum computing research focuses on the development of efficient quantum algorithms and quantum programming languages that should allow to code such algorithms (see, e.g., [10]). Pursuing this route has the advantage of directing the efforts in identifying the complexity classes including all the problems that can be efficiently solved in the quantum setting (e.g., the classes QP and BQP). However, it requires a deep change in the way we instruct new generations of computer scientists on algorithms and programming. The risk is that of finding only “small” classes of problems over which the quantum advantage is substantial.

This paper follows an alternative path: coding quantum algorithms in a classical setting, relying on a compiler to achieve quantum speed-up. Ideally, the programmer would thus be relieved from any "quantum mechanical duty." However, the generalized speed-up provided by a compiler would hardly compare with the speed-ups available for each given problem.

In light of these observations, a compromise between the two approaches seems reasonable. Though, two questions arise: (i) What kind of compiler? (ii) What kind of classical paradigm?

In Section 2 we recall the tight connection between graphs and procedural models (e.g., imperative programming languages). Throughout Section 3, the analogy between quantum computational and graph reversibility frequently emerges in a rather natural way. However, graphs defined by the operational semantics of procedural models are almost never reversible, being termination states sink vertices. We saw that whenever lacking graph reversibility, projectors play a fundamental role in providing quantum representation of graphs. However, to rely on projectors means to give up on unitary evolution. So it seems as if procedural computations and unitary evolution are like oil and water.

One could argue that a reversible operational semantics can be defined for any imperative program that always terminates. Which is of course true, as it is true that classical Boolean circuits can be embedded into reversible ones, and hence into quantum circuits. The cost must be paid in terms of space occupancy and an upper bound for the number of computational steps should be available a-priori in order to allocate the right amount of space (qubits). Notice that a proposal for a compiler of a fragment of the C language into Quantum Annealers has been presented in [21]. It is worth mentioning that even though Quantum Annealers are currently one of the most used architectures that exploit quantum mechanics, they are not general quantum machines and the debate on the speed-up they achieve is still open (see, e.g., [22, 23]). As a matter of fact, the most recent works in the direction of defining the boundaries of *quantum supremacy* rely on photonic processors [24].

We argue that a better chance for compiling classical languages into quantum comes from declarative languages. The most easy way to get a grasp of our intuition is that of considering assignments and the no-cloning theorem. The no-cloning theorem states that the function that makes a copy of an unknown, generic quantum state is not unitary, and as such it cannot be realized in a purely quantum model. While this is a major obstacle for the compilation of procedural languages, where assignments make copies of values, it is coherent with the spirit of declarative programming, where assignments and in general side effects are avoided. We argue significant developments on this matter may come from joint efforts between the two involved research fields.

In the case of functional languages, the most natural semantics is the denotational one, which associates to a program the *function* it defines. Since classical architectures are by nature operational, the denotational semantics of such languages have been mainly exploited to prove correctness of the programs, while the effort of producing the equivalent procedural version is paid by the compiler. In this sense a quantum compilation would instead consist in the embedding of the denotation of the program into a *reversible unitary function*, which should then be compiled into a quantum circuit. The compilation of unitaries into quantum circuits is a hot topic in the quantum community and different techniques are under development to optimize the task (see, e.g., [25, 26]). As far as the embedding of a computable function into a unitary one is concerned, since the framework of Quantum Circuits becomes Turing Complete only when we endow it a classical model that generates *uniform families of circuits*, we cannot expect to be able to compile any computable function into a unitary one. This is the point where we envisage that *compositionality* properties of the denotational semantics should come into play to decompose the whole denotation in subcomponents. For each of these subcomponents the compiler should be able to identify whether it is amenable to quantum compilation and quantum speed-up or it is better to implement it in a classical fashion.

As far as logic and constraint based languages are concerned, a similar analysis should be conducted referring to *model-theoretic semantics* instead of denotational ones. In a sense this makes the picture more challenging than the functional case. It comes not as a surprise being the framework the most high level one. However, our analysis of graphs and quantum walks is pertinent with several aspects of this approach.

One concerns the solving stage in *Answer Set Programming (ASP)*, which relies on a dependency graph for the assignment of truth values to the atoms, in accordance with the constraints defined in the model. Dependency graphs supporting stable models and quantum walk amenable graphs share few commonalities due to loops. Future studies could focus on encoding techniques to conciliate these two conflicting classes of graphs.

The link between ASP and quantum computation is reinforced by the work in [27], where stable models search was given a solution through Grover's algorithm. Aside from the resulting quadratic speed-up, the work leads to a broader implication: any ASP program may be executed on quantum architecture. This is a first effort in the same direction that we are proposing; our aim is to generalize it. Ideally, the investigations could set off from quantum SAT solvers [28], thus extending the above result to any SAT solving-based program.

The case of MiniZinc is analogous: a graph underlies the evolution of the domains of the variables. In this context, *consistency properties* over such graph are at the basis of the computation. While on classical machines many techniques have been developed to remove symmetries over the graph, thus reducing the computational complexity, in the case of quantum computation we observed in Section 3 that the more a graph is *symmetric* the more likely it is amenable to quantum encoding and quantum walks.

References

[1] R. P. Feynman, Simulating physics with computers, in: Feynman and computation, CRC Press, 2018, pp. 133–153.

[2] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Review 41 (1999) 303–332. doi:10.1137/S0036144598347011.

[3] L. K. Grover, A fast quantum mechanical algorithm for database search, in: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96, Association for Computing Machinery, New York, NY, USA, 1996, p. 212–219. URL: https://doi.org/10.1145/237814.237866. doi:10.1145/237814.237866.

[4] L. Euler, Solutio problematis ad geometriam situs pertinentis, Commentarii Academiae Scientiarum Imperialis Petropolitanae 8 (1736) 128–140.

[5] C. H. Papadimitriou, Computational complexity, 1993.

[6] A. Montanaro, Quantum walks on directed graphs, arXiv preprint quant-ph/0504116 (2005).
[7] J. Kempe, Quantum random walks: An introductory overview, Contemporary Physics 44 (2003) 307–327. doi:10.1080/00107151031000110776.
[8] A. S. Bhatia, A. Kumar, Quantum finite automata: survey, status and research directions, 2019. arXiv:1901.07992.
[9] A. W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Physical review letters 103 (2009) 150502.
[10] M. Ying, Foundations of Quantum Programming, 1st ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2016.
[11] M. A. Nielsen, I. Chuang, Quantum computation and quantum information, 2002.
[12] F. Harary, Graph Theory, Addison Wesley series in mathematics, Addison-Wesley, 1971.
[13] R. Motwani, P. Raghavan, Randomized algorithms, Cambridge university press, 1995.
[14] D. Aharonov, A. Ambainis, J. Kempe, U. Vazirani, Quantum walks on graphs, in: Proceedings of the thirty-third annual ACM symposium on Theory of computing, 2001, pp. 50–59.
[15] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, J. Watrous, One-dimensional quantum walks, in: Proceedings of the thirty-third annual ACM symposium on Theory of computing, 2001, pp. 37–49.
[16] A. M. Childs, Universal computation by quantum walk, Phys. Rev. Lett. 102 (2009) 180501. doi:10.1103/PhysRevLett.102.180501.
[17] N. B. Lovett, S. Cooper, M. Everitt, M. Trevers, V. Kendon, Universal quantum computation using the discrete-time quantum walk, Physical Review A 81 (2010). doi:10.1103/physreva.81.042330.
[18] S. Severini, On the digraph of a unitary matrix, SIAM Journal on Matrix Analysis and Applications 25 (2003) 295–300. URL: https://doi.org/10.1137%2Fs0895479802410293. doi:10.1137/s0895479802410293.
[19] S. Severini, Graphs of unitary matrices, 2003. URL: https://arxiv.org/abs/math/0303084. doi:10.48550/ARXIV.MATH/0303084.
[20] D. Della Giustina, C. Piazza, B. Riccardi, R. Romanello, Directed graph encoding in quantum computing supporting edge-failures, in: C. A. Mezzina, K. Podlaski (Eds.), Reversible Computation, Springer International Publishing, Cham, 2022, pp. 75–92.
[21] M. W. Hassan, S. Pakin, W.-c. Feng, C to d-wave: A high-level c compilation framework for quantum annealers, in: 2019 IEEE High Performance Extreme Computing Conference (HPEC), IEEE, 2019, pp. 1–8.
[22] M. H. Amin, Searching for quantum speedup in quasistatic quantum annealers, Physical Review A 92 (2015) 052323.
[23] J. King, S. Yarkoni, J. Raymond, I. Ozfidan, A. D. King, M. M. Nevisi, J. P. Hilton, C. C. McGeoch, Quantum annealing amid local ruggedness and global frustration, Journal of the Physical Society of Japan 88 (2019) 061007.
[24] L. S. Madsen, F. Laudenbach, M. F. Askarani, F. Rortais, T. Vincent, J. F. Bulmer, F. M. Miatto, L. Neuhaus, L. G. Helt, M. J. Collins, et al., Quantum computational advantage with a programmable photonic processor, Nature 606 (2022) 75–81.
[25] P. Niemann, R. Wille, R. Drechsler, Advanced exact synthesis of clifford+t circuits, Quantum

Information Processing 19 (2020). doi:10.1007/s11128-020-02816-0.

[26] M. Soeken, M. Roetteler, N. Wiebe, G. D. Micheli, Logic synthesis for quantum computing, 2017. arXiv:1706.02721.

[27] D. Meyer, J. Pommersheim, J. Remmel, Finding stable models via quantum computation., 2004, pp. 285–291.

[28] M. Mosca, S. R. Verschoor, Factoring semi-primes with (quantum) sat-solvers, Scientific Reports 12 (2019).