Quantum Spatial Computing

Martin Werner
Institute for Applied Computer Science & Research Institute CODE,
Bundeswehr University Munich, Germany
martin.werner@unibw.de

Abstract

Quantum computing is expected to create a major shift across the whole computing industry given that it can solve a certain set of operations that used to be very hard in comparably short time. In this article, the author intends to introduce central aspects of quantum algorithms to the GIS community and explores some probable directions of research related to quantum computing for spatial algorithms. The author hopes that this paper enables researchers of our community to place an informed decision whether and when it is worth looking at this exciting new area of algorithm research and computing while staying on an abstraction level high enough for concise presentation.

1 Introduction

In the last decade, quite a few companies are having success with building first real-world quantum computers. Among these are D-Wave systems [6], Google [10], and IBM [12]. However, these machines are currently very small and their usage implies lots of noise. Nevertheless, quantum computers are nowadays existing and there is a race of growing them to a useful size. The interesting aspect of quantum computing is that it can provide exponential speedups in certain situations. For example, it is possible to factor integer numbers in polynomial time using Shor's algorithm on a quantum gate computer or it is possible to solve certain NP-hard problems in polynomial time.

In this paper, we explore how quantum computing could affect the spatial computing domain in a very broad sense and as the community might not have been exposed too much with basic concepts of quantum computing, the paper will first briefly recall the computational models that quantum computers operate in, because these form a meeting point between the engineers and physicians actually trying to build computers and the computer scientists looking for algorithms that can exploit quantum computing peculiarities.

2 Quantum Computing

Quantum computing is an idea in which quantum effects are being used to model information. Similarly to digital computers, the atomic unit of information is a bit, called qubit in the quantum computing domain. Similarly to the situation of a digital computer, a qubit can typically be set to two different values (e.g., true and false) and if it is read, it will realize a binary value (true or false). Internally, however, it can be in a superposition state where it actually takes "both values at the same time". If a quantum bit is in perfect superposition and you read it, you will get a zero and a one with equal probability. When introducing quantum gate computers in a later section, we will understand much more about qubits.

From an application perspective, the simplest form of quantum computing is adiabatic quantum computing or quantum annealing. A quantum annealer is a specialized quantum computers that is built to solve a certain type of optimization problem using qubits that are in superposition and entangled with each other according to a problem specification. As we want to avoid explaining the physics and engineering of building quantum annealers, we will describe the computational model only, which is the model of Quadratic Unconstrained Binary Optimization (QUBO). This model is equivalent to the (physics-inspired) Ising model, you can jump back and forth with simple algebraic modifications.

A QUBO is given as a quadratic form x^tAx where $x \in \mathbb{B}^n$ is a vector of binary values and A is a matrix of real numbers. The QUBO task is to find the vector $x \in \mathbb{B}^n$ that minimizes this quadratic form given that x is binary. As a rough idea of how a physical implementation works, let us just say that it is based on the energy minimization principle of nature. That is, the matrix is somehow encoded into some hardware and then the hardware is cooled down near to the absolute zero point of temperature and then the bits can be read that minimize the QUBO because the solution to the QUBO minimizes the energy of the physical construction used. This is essentially how the D-Wave quantum annealer works. That is, from an application perspective (e.g., as a spatial computing researcher), you will have to encode your problem into the QUBO formulation, let the quantum annealer solve your QUBO and expect the results to be very noisy or, in other words, your algorithm should be able to deal with good yet suboptimal solutions to the given QUBO. Usually, quantum computers are used over the Internet, because operating them is a very difficult task (e.g., cooling down as much as possible, minimizing all sorts of noise influence to the machine, etc.). Hence, even more hands-on, you would prepare a QUBO matrix, send it to the annealer and get a vector back that might be good in terms of minimizing the QUBO form.

The second and more advanced form of quantum computing is a quantum gate computer. To (informally) explain a quantum gate computer, we need to go back to the concept of a qubit. In fact, a qubit is represented as the tensor product of two one-dimensional complex vector spaces. It is customary to introduce two symbols $|1>$ and $|0>$ to represent two basis elements of this tensor product. That is, $|0>$ can be identified with the vector $(1, 0)$ and $|1>$ with $(0, 1)$ Then a qubit is given by two complex numbers α and β formling a linear combination

$$\alpha|0> + \beta|1> \text{ constrained by } |\alpha|^2 + |\beta|^2 = 1.$$

This allows for a geometric interpretation as the topological space of a qubit is homeomorphic to the surface of a sphere in $\mathbb{R}^3$ known as the Bloch sphere. As quantum gate computers come with a preferred basis which provides the link to boolean values (e.g., a quantum $|1>$ represents true, a qubit $|0>$ represents false, all other qubits represent true and false at the same time with varying probability), quantum gates are given as 2x2 matrices with complex entries. These matrices represent rotations, thus, need to be unitarian. This is an interesting aspect as this implies that quantum gate computers are reversible computers, e.g., each operation you can do can be undone as well.

Here are some widely-used gates, namely the N gate which provides negation, the Hadamard gate H which brings basis elements into "perfect superposition". The V and Z gate operate only on the $|1>$ part leaving $|0>$ unchanged, where $V_\pi = Z$.

$$N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, V_\theta = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

It is customary to look at the Hadamard gate once: Operating H on $|0>$ gives

$$H|0> = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0> + |1>)$$

The magnitudes of α and β now give the probability that a measurement of the qubit returns true ($|\alpha|^2$) or false ($|\beta|^2$). With this information, it is easy to see that measuring $H|0>$ returns true and false with a probability

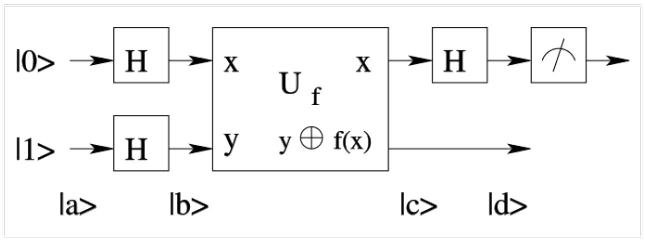

Figure 1: Two-qubit implementation of quantum algorithm of Deutsch.

of 50% each. Given that there are infinitely many unitarian matrices that could be made a quantum gate in a physical quantum computer implementation, it is worth noting that there are universal sets of gates with which any unitarian matrix can be produced up to some given error. One such set is given by H, N_C (the application of some generic procedure on N), and $V_{\frac{\pi}{4}}$. For details, the reader is referred to the excellent tutorial [19].

Now, these quantum gate computers usually bind together a few qubit into a quantum register and the gates operate on register level. The first non-trivial algorithm is given by the following quantum algorithm which is able to decide whether two things are equal by only reading once. This is a powerful concept and actually motivates the Shor algorithm for factorizing large integer numbers.

The basic process of applying a quantum gate computer to a problem is now to first set a certain quantum register to a known base-state (e.g., combinations of the base states $|0>$ and $|1>$ per qubit). Then, some operations bring some of the qubits in superposition such that the following operations can operate on them. Finally, the qubits should have been modified by the algorithm that sampling from them reveals the answer to the given problem.

One widely-used trick here, valid for any boolean function f, is the construction of U_f, which given a Boolean function

$$f : \mathbb{B}^m \to \mathbb{B}^n$$

provides a unitarian map

$$\hat{f} : \mathbb{B}^m \times \mathbb{B}^n \to \mathbb{B}^m \times \mathbb{B}^n$$

by sending $(x, y) \to (x, y \oplus (fx))$ [1].

This is, for example, used in the most simple non-trivial quantum algorithm known as the algorithm of Deutsch [9]. This can be used to decide, whether a given one-dimensional Boolean function $f.\mathbb{B} \to \mathbb{B}$ is constant, e.g., $f(0) = f(1)$, or balanced, e.g., $f(0) \neq f(1)$. In classical computing, it is possible to do this by reading both $f(0)$ and $f(1)$, but not faster. For a quantum computer, however, it is possible to do so faster.

To understand this algorithm, one can just compute it operation by operation following the diagram in Figure 1 [2]. While you are doing so, you realize that f is evaluated exactly once but using a qubit in superposition and that the algorithm returns the needed information about f, which would be classically impossible. Assuming that the evaluation of f takes, say, one year, this brings runtime from two years in the classical model down to one year using a quantum computer.

This is possibly the simplest situation in which one can see how and that Quantum algorithms can be significantly faster for traditional problems formualted in Boolean functions and concludes our short trip of quantum gate computers.

[1] In this context, $\oplus$ is used for addition modulo 2

[2] The image has been taken from Wikipedia, cf. `https://de.wikipedia.org/wiki/Deutsch-Jozsa-Algorithmus`

3 Quantum Computing Examples

This section provides a few examples and resources on existing quantum algorithms with a tie to spatial computing. However, it is not intended to be complete or to introduce any of these solutions, it shall rather provide a concise starting point for interested readers to explore the field.

3.1 Logical Networks of Light Switches

Adiabatic quantum computing is often introduced with the light switch game. Given a network of n light switches connected with some logics to a single lamp, find the setting in which the light is turned on without knowing the connectivity network of the light switches. In fact, this is a hard problem which could practically only be solved with a brute force approach or additional information. A brute force approach, however, needs to evaluate 2^n possible states of the n switches and becomes intractable even for moderate n.

As a good source for learning this problem, we refer the user to the more general class of cirquit satisfiability problems [18] and to the D-Wave introduction to their quantum computer [5].

3.2 Map Coloring

The map coloring problem is a nice problem for understanding how an adiabatic quantum computer works in a spatial setting and it is also given a core example in the D-Wave manual, because it is conceptually easy yet shows the limitations of the D-Wave hardware (namely that there aren't couplers between all qubits) and how to circumvent this by "duplicating" qubits constraining them to be equal. The central question when doing spatial computing with quantum computers will be what a bit represents. Traditionally, spatial computing takes place a lot in a geometry domain with floating point representations of geometry. This domain is not well-suited for adiabatic quantum computing, because it is not easy to find what a qubit should represent. Topology in contrast provides many problems that are of combinatorial nature and, therefore, very well-suited for adiabatic quantum computing.

The map coloring problem aims to assign colors to a planar map such that no two neighbors have the same color. In the mathematics, this problem has been widely discussed, because it remained unclear and is considered the first real mathematical problem of relevance that has been solved using a computer. In 1852, Francis Guthrie formulated the conjecture that any given planar map can be coloured with four colors while creating a map of England. After that, a sequence of attempts in proving this conjecture has been established where most solutions survived a decade, but finally were proven wrong. During this time, however, an upper bound with five colors was established by Heawood. Heinrich Heesh proposed a computer proof which has never been implemented due to limited machine power, however, it was refined a few times bringing down the set of solutions which need manual or computer checking to 1936 (Appel and Haken, 1976) leading to a publication with 400-page appendix in 1989. Finally, this problem was formally proven in 2005 using the computer-assisted proof environment Coq by Benjamin Werner (still not closing the debate in mathematics, because hardliners accept proofs only if they are fully transparaent in the sense that they are accessible to the mind of humans).

How can this be implemented in a quantum computer? As said, the pressing question is how to map the problem to qubits. And in this case, it can be done as follows: Each country in the map is assigned a color represented as a one-hot encoding of qubits. That is, we assign a set of k qubits to each country in the map and constrain them such that exactly one of those qubits is one at a time. Then, the topology of neighbors is encoded as constraints, essentially, that their vectors must not be the same. This can be done in the QUBO framework and it can be mapped to the D-WAVE hardware if the number of qubits suffices. There is a D-WAVE white paper with lots of details on this problem [7].

3.3 Dynamic Time Warping

Dynamic Time Warping is another very nice example showing how geometric problems can be supported by quantum computers. Dynamic time warping is a distance definition which has found many applications. It has been applied to varying application areas including speech recognition [20], handwriting recognition and signature verification [4], in computer vision [1], in shape retrieval [16], in computational biology and medicine [15], in pattern recognition [2], and recently similarity search for spatial trajectories [21].

Dynamic Time Warping is based on extending the idea of a point-wise Euclidean distance of time sreies by allowing the "matching" to change according to dynamic programming rules. In fact, the dynamic time warping distance of trajectories (and time series) is defined to be the sum of the distances of points of both trajectories, where the points are matched based on minimizing over all matchings. A matching, in this context, is an assignment of points of one trajectory with points of the other trajectories. For dynamic time warping, a matching always starts by matching the first two points of the trajectory. Then, it can either go one step in one of the trajectories or one step in both. Ulitmately, however, it has to reach the last vertex of both trajectories. This is a natural dynamic programming problem and can be formulated as follows [22]:

$$d_{\text{DTW}}(a_{1\dots n}, b_{1\dots m}) = \begin{cases} 0 & \text{if } a \text{ and } b \text{ are empty} \\ \infty & \text{if only one of } a \text{ and } b \text{ is empty} \\ d(a_n, b_m) + & \min \begin{cases} d_{\text{DTW}}(a_{1\dots n-1}, b_{1\dots m-1}) \\ d_{\text{DTW}}(a_{1\dots n-1}, b_{1\dots m}) \\ d_{\text{DTW}}(a_{1\dots n}, b_{1\dots m-1}) \end{cases} \end{cases}$$

This distance can be computed in $O(mn)$ time [8]. It is usually composed of first computing the full distance matrix and, then, searching for the shortest path in this distance matrix going right, up, or both at the same time.

We will now simplify to dynamic time warping problems given with equal-length trajectories of n points each. As a quadratic optimization problem, DTW can be written using the distance matrix and a set of constraints only. Concretely, consider the set $\mathcal{A}_{m,n}$ of monotonous alignment matrices. These are binary matrices containing a path from the upper-left corner (1,1) to the lower-right corner (m-1,n-1) containing only of moves $\rightarrow$, $\searrow$, and $\downarrow$. The size of this set is known as the Delannoy number, cp. sequence A001850 in the OEIS. With this set in place, the DTW distance is given as

$$d_{\text{DTW}}(x, y) = \min_{A \in \mathcal{A}_{n,n}} < A, \Delta(x, y) >, \tag{1}$$

where $< -, - >$ is the Frobenius inner product of matrices.

But how can we map this to a quantum computer? One approach is to write the expression using binary vectors $x \in \mathbb{B}^{n^2}$ obtained by flattening the path matrices exploiting the fact that the Frobenius product is the sum of the entries of the Hadamard product of the two given matrices.

$$d_{\text{DTW}}(x, y) = \min_{x = \text{flatten}(A) \text{ for } A \in \mathcal{A}_{n,n}} x^t D x$$

This is now clearly a quadratic optimization problem, but how can we ged rid of the constraint? It is possible to represent certain constraints as matrices as well. Such a matrix C would, for example, take values

$$x^t C x = \begin{cases} 0 & \text{if } \operatorname{mat}(x) \in \mathcal{A}_{n,n} \\ 1 & \text{else,} \end{cases}$$

where mat denotes the operation turning the vector x back into a square path matrix. Assuming such a matrix C exists, DTW could be solved by solving

$$\min_x x^t D x + \lambda x^t C x = \min x^t (D + \lambda C) x \tag{2}$$

If λ is chosen large enough (e.g., $2n \max D$), then each time the constraint matrix would fire, the value in Eq. 2 would be impossible to become smaller than the worst value one could generate with the distance matrix part. If, however, the constraint is fulfilled, we are left with the situation of DTW in Eq. 1.

This concludes our meta-algorithm for DTW. The art of programming adiabatic quantum computers is now the come up with constraints that are good enough to guarantee an optimal solution. Note that for the case of DTW, a matrix C with the properties above cannot exist, because $x = 0 \in \mathbb{B}^{n^2}$ is clearly not a path matrix, yet $x^t Ax$ is zero. Therefore, and this is the heart of QUBO-based quantum computing, we have to relax from the case of ideal constraints and come up with alternatives that allow for concluding the final result or parts thereof.

3.4 Quantum Gate Computing Strategy

Quantum gate computing allows for solving a larger set of problems as opposed to the adiabatic case, however, it is not as accessible. However, quantum gate computers can solve many of the NP-hard combinatorial problems as well and go beyond that. A nice overview of existing algorithms for quantum gate computers is maintained online in the Quantum Algorithm Zoo [13].

If you want to exploit the power of a quantum gate computer in spatial computing, you have to design what each bit will represent and at the same time, you have to prepare an intuition of what a superposition of these bits can bring to you and how quantum gates will encode the constraints of the solution to your problem at hand. This domain is still in its infancy as it is not very easy to come up with intuitions like that. Therefore, for spatial computing, I expect quantum gate computers to be used more often in an indirect way: *If you want to apply a quantum gate computer to spatial computing and don't have an innovative intuition on how to exploit and modify bits in superposition, you will still be able to rely on the solution capabilities of quantum computers for known quantum algorithms.*

For a quantum computer, this includes the solution of traveling salesman type problems as well as QUBOs. That is, one common way of "programming" a quantum gate computer might be by decomposing the spatial problem into a set or sequence of hard problems for which a quantum algorithm has already been proposed.

There are plenty of examples of how to formulate spatial optimizations in form of traveling salesman type problems and it is left to the reader to exercise with this philosophy.

3.5 Quantum Machine Learning

As quantum computers are very fast in solving certain types of optimization, they are a candidate for many algorithms in the machine learning domain that internally rely on optimization. One point in time where this journeys start might be seen is in the paper [3]. A major milestone in the sequel is Harrow, Hassidim, and Lloyd's quantum algorithm for solving linear systems [11]. Lately, Kerenidis and Prakash's algorithm for recommender systems provided exponential speedup over any known classical algorithm [14]. Most interestingly, this algorithm has given the intuition to a classical algorithm with an equivalent asymptotic runtime complexity [17]. This is a nice story for finishing this short introduction to quantum computing as it makes clear that understanding quantum algorithms can directly feedback to the field of classical algorithms and is tightly related to a family of algorithms nowadays known as randomized algorithms, which are, as well, underrepresented in the spatial computing community of these days.

4 Research Need

Spatial computing was mostly inspired by the geometry of the things we discuss. It is quite typical that this happens in a continuous representation of space using floating point numbers though there are a few approaches that try to solve spatial computing problems in an integer domain (graph rounding, fixed-precision arithmetics, binarization). In the last decades, the topology of things is being used more and more in spatial computing and methods from topology quite naturally generate combinatorial problems. The map coloring problem is a typical example where the topology relations of neighboring polygons is representing the spatial aspect and the resulting optimization problem is combinatorial.

Given that quantum computers will remain small for a certain time, the spatial computing community should look into how certain problems can be solved in a combinatorial or at least in an integer-rounded fashion in order to make efficient use of individual qubits.

In addition, the spatial computing community was used to solve their problems on their own by extending and adopting solutions to similar non-spatial problems. The author is convinced that there are quite a few spatial computing problems that can be solved by using the quantum computer as an oracle solving a certain set of hard problems. In this setting, quantum algorithm research takes the form of finding problem transformations and problem reformulations such that a given problem of interest is expressed as a sequence or low-complexity algorithm formulating how the problem should be solved under the assumption that a set of hard problems has a quantum computing algorithm of small time- and space complexity.

5 Conclusion

We are living in a time in which the first commercial quantum computers reach the market. At the same time, there has been a lot of research in the area of quantum computing which motivated to build a quantum computer. The community is centered around proving "quantum supremacy" in many senses, namely, that the expensive research on quantum computers and the expensive operation of those is needed, because there are problems intractable for classical computers that become solvable. In consequence, research that uses quantum computers on problems that are not intractable today is in its infancy and an exciting area for further research. As an example, the dynamic time warping distance has been heavily researched in the classical computing model but is typically used in a restricted setting where the amount of warping is limited in order to make it both indexable and fast. With a quantum computer and certain developments that are beyond the scope of this paper, we might hope to remove this typical restriction and, thereby, allow dynamic time warping to be used without such warping restriction in a scale-free manner.

With this paper, the author hopes to have motivated a few spatial computing researchers to consider quantum computing in future research by looking for instances of hard problems that you are currently avoiding with tricks but quantum computer could directly solve as well as for problems with a natural structure of binary optimization.

References

[1] A. Almog, A. Levi, and A. M. Bruckstein. Spatial de-interlacing using dynamic time warping. In *IEEE International Conference on Image Processing*, volume 2, pages 1006–1010. IEEE, 2005.

[2] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[3] N. H. Bshouty and J. C. Jackson. Learning dnf over the uniform distribution using a quantum example oracle. *SIAM Journal on Computing*, 28(3):1136–1153, 1998.

[4] W.-D. Chang and J. Shin. Modified dynamic time warping for stroke-based on-line signature verification. In *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 724–728. IEEE, 2007.

[5] Quantum computing primer. https://www.dwavesys.com/tutorials/background-reading-series/quantum-computing-primer.

[6] D-WAVE Systems, 2019. https://www.dwavesys.com/.

[7] E. D. Dahl. Programming with d-wave: Map coloring problem, 2013.

[8] K. Deng, K. Xie, K. Zheng, and X. Zhou. Trajectory indexing and retrieval. In *Computing with spatial trajectories*, pages 35–60. Springer, 2011.

[9] D. Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[10] Google AI Quantum, 2019. https://ai.google/research/teams/applied-science/quantum/.

[11] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15), 2009.

[12] IBM Q, 2019. https://www.ibm.com/quantum-computing/.

[13] S. Jordan. Quantum Algorithm Zoo, 2019. https://quantumalgorithmzoo.org/.

[14] I. Kerenidis and A. Prakash. Quantum recommendation systems. *arXiv preprint arXiv:1603.08675*, 2016.

[15] B. Legrand, C. Chang, S. Ong, S.-Y. Neo, and N. Palanisamy. Chromosome classification using dynamic time warping. *Pattern Recognition Letters*, 29(3):215–222, 2008.

[16] A. Marzal, V. Palazón, and G. Peris. Contour-based shape retrieval using dynamic time warping. In *Current Topics in Artificial Intelligence*, pages 190–199. Springer, 2005.

[17] E. Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 217–228. ACM, 2019.

[18] A. S. Tannenbaum. NP-hard problems. 2014.

[19] B. Valiron. Quantum computation: a tutorial. *New Generation Computing*, 30(4):271–296, 2012.

[20] V. Velichko and N. Zagoruyko. Automatic recognition of 200 words. *International Journal of Man-Machine Studies*, 2(3):223–234, 1970.

[21] H. Wang, H. Su, K. Zheng, S. Sadiq, and X. Zhou. An effectiveness study on trajectory similarity measures. In *Proceedings of the Twenty-Fourth Australasian Database Conference-Volume 137*, pages 13–22. Australian Computer Society, Inc., 2013.

[22] B.-K. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proceedings of the 14th International Conference on Data Engineering*, pages 201–208. IEEE, 1998.