EPJ Web of Conferences **274**, 09002 (2022) https://doi.org/10.1051/epjconf/202227409002
XVth Quark Confinement and the Hadron Spectrum

Vertex reconstruction in particle detectors using quantum computing algorithms.

Francisco Matorras Cuevas[1,*] and *Pablo* Martínez Ruiz del Árbol[1,2,**]

[1]Universidad de Cantabria
[2]Instituto de Física de Cantabria, UC-CSIC

Abstract. This work aims at testing a new quantum computing algorithm to reconstruct vertices in the context of a tracker-like particle detector. Input tracks have been generated using a simplified tracker simulator assuming they originate from two different vertices. The Point of Closest Approach of the tracks to the beam line has been considered as the nodes of a graph connected to the other nodes through a weight equivalent to the euclidean distance between the two. A Variational Quantum Eigensolver algorithm has been used in order to divide the graph in two groups that maximizes the total distance between the two groups. The algorithm has been implemented using Qiskit, the IBM framework, obtaining a track-vertex association accuracy of about 90% for distances between vertices of a few milimeters. This work represents a simple proof-of-concept that a quantum computing algorithm can be used to solve the problem of the vertex reconstruction.

1 Introduction

Quantum technologies are becoming more and more popular in the last years, attracting the attention of companies, universities, and scientists. This topic encompasses different individual branches, such as quantum sensors, cryptography, simulation, imaging, and computing. The latter, quantum computing [1, 2], is particularly interesting because it offers a new paradigm in which many mathematical problems of great computational complexity can be solved in affordable times.

The basic difference between classical and quantum computing is connected to the type of basic information unit used in each paradigm. While classical computing uses bits, which can either be "on" or "off" (or 0 and 1), quantum computing makes use of the so called quantum bits (qbits). These take advantage of quantum superposition, and are able to be in states which are combination of 0 and 1. Or, to be more precise, of $|0\rangle$ and $|1\rangle$. These new objects can be engineered, using its quantum properties, in order to find solutions to interesting mathematical problems, typically of combinatorial nature. Some of the most famous examples of a problem with these characteristics are the factorization of large prime numbers and the properties of atoms and molecules of systems of more than three atoms [3].

*e-mail: francisco.matorras@alumnos.unican.es
**e-mail: parbol@ifca.unican.es

This paper aims at implementing a quantum computing algorithm to perform the vertex reconstruction and identification in the context of Particle Physics collision experiments.

2 Quantum Computing

The key concept in a quantum computer is the qbit. The qbit is a physical quantum system that can be in a combination of two different states, or, to be more precise, it can be mathematically described by a two-dimensional Hilbert space [1, 2].

$$|0\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{1}$$

Therefore, any quantum state of the system can be described as the superposition of the two elements of the basis.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2}$$

In Equation 2 the amplitudes are complex, and only need to fulfil the normalization requirement. That means the sum of the squares must equal 1. Consequently, the amplitudes may be expressed in terms of two angular variables, the example we use is that $\alpha = \cos\theta/2$ and $\beta = e^{i\phi}\sin\theta/2$. This parameterization allows to express the qbit state as a point in the surface of the Bloch sphere [4].

2.1 Circuit model

A classical computer only allows its bits to be in a binary state. Therefore, the total state for a system formed by n number of bits would have 2^n possibilities. However, a system the same size realized on a quantum computer, with n number of qubits, instead, would have a 2^n dimensional Hilbert space. This characteristic provides a considerably larger computational capacity to quantum computers compared to the one their classical counterparts have.

The modern paradigm of quantum computing is based in the concept of quantum circuits. A quantum circuit is a set of qbits transformed by a set of operators that are applied sequentially[1, 2]. These operators are referred to as quantum logic gates, by comparison to logic gates in classical computing. Quantum circuits transform an initial wave function, represented by the set of quantum bits, into another wave function that might be useful for a given problem. Another way of explaining the logic gates and the circuits in general, is to see them as rotations in the aforementioned Bloch sphere. Among some of the most popular quantum logical gates, the CNOT, Hadamard or AND gates can be found.

2.2 The VQE algorithm

The Variational Quantum Eigensolver (VQE) [5] is a well-known quantum algorithm that has been used to solve many different optimization problems. It belongs to a class called "hybrid quantum algorithms", as it uses a quantum computer to encode the problem into a wave-function that depends on a number of parameters, that is later used to evaluate a Hamiltonian; but, in addition to this, it uses a classical minimizer, to find the set of parameters that minimize the value of $\langle\psi|H|\psi\rangle$. The minimum of the average value of the

Hamiltonian is obtained when the wave-function corresponds to the eigenstate with the lowest eigenvalue. In this way, the VQE algorithm is able to find this eigenstate. Figure1 shows a diagram with a description of the algorithm.

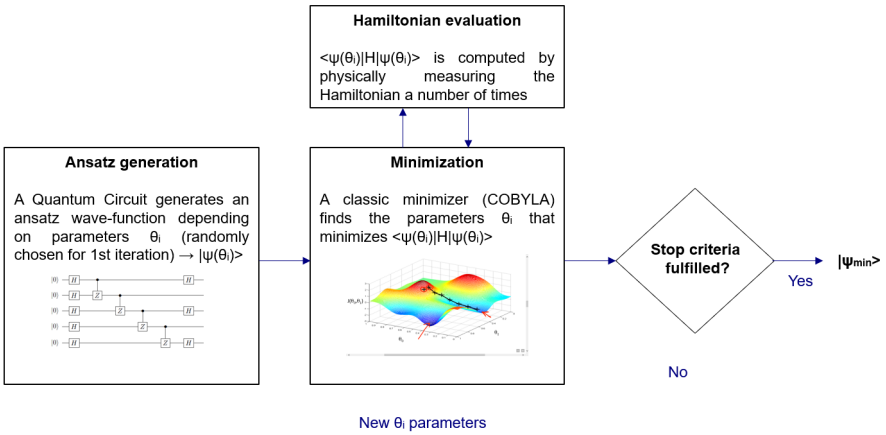


Figure 1: Diagram that shows the steps VQE algorithm take and their description.

The first step of the algorithm consists in the creation of an *Ansatz* wave-function. This function is formed by a set of qbits transformed through a circuit of quantum gates. The parameters of the quantum gates will be free parameters of the problem, in such a way, that the final Ansatz will depend on them. In a second step, the Hamiltonian will be evaluated using the Ansatz function. This will be done a number of times, referred to as the number of shots. The result of this evaluation will be used by a classical minimizer in order to find the parameters of the ansatz function that minimize the measured value of the Hamiltonian. The VQE algorithm frequently uses a minimization algorithm known as, COBYLA [6], which stands for "Constrained Optimization BY Linear Approximation". The working principle of COBYLA relies on an iterative procedure in which the problem is linearized and solved until a good enough solution is found.

3 The vertexing problem and VQE

Vertexing is a common problem in particle collider experiments where several interaction vertices may exist for every bunch crossing of the beams. Most of these vertices are of limited interest and only contribute by adding spurious tracks that increase the difficulty of studying more interesting vertices. In order to clean the environment from these spurious tracks, reconstructed tracks are clustered and associated to a vertex. This process is known as vertex reconstruction. Modern particle experiments use classical clustering algorithms such as k-means [7] or deterministic annealing [8], among others, in order to perform this task.

3.1 Vertexing as a Max-Cut problem

The vertex reconstruction problem with a total of two vertices can be formulated in terms of the so-called Max-Cut problem in graph theory. Let there be a set of edges and vertices

(mathematical vertices, and not the particle physics ones) connecting them. The simplest version of the Max-Cut problem can be expressed as follows: having some of the vertices in the graph connected by a number of edges, how could the vertices be divided into two different subsets (or groups) so that the number of connecting edges between the subsets is maximized. A more sophisticated definition of the problem, considers that edges have weights assigned to them, in such a way that the figure of merit is the weighted number of connecting edges.

This mathematical problem may be expressed as a function F, that consists of sums, this is the max cut characteristic function.

$$F = \sum_i \sum_j x_i Q_{ij}(1 - x_j) \tag{3}$$

Here the values $x_{i,j} \in [0, 1]$ represent the subset the vertex belongs to. For instance, in a graph of 5 vertices numbered from 1 to 5, the combination $(0, 0, 1, 0, 1)$ indicates that vertices 1, 2 and 4 belong to the same subset or group, while vertices 3 and 5 belong to the other one. With this reasoning, only in the case when they are part of different groups, they contribute to the total sum. The term Q_{ij} is a matrix where the weight between every two vertices is stored. The diagonal will be zero (as the distance from a vertex to itself will always be zero). The combination of x_i that minimizes this function is the solution of the Max-Cut problem.

Many Quantum Computers implement Hamiltonians of the form:

$$H = \sum_{i,j} \sigma_i^{(z)} \sigma_j^{(z)} \omega_{ij} + \sum_i \omega_i \sigma_i^{(z)} \tag{4}$$

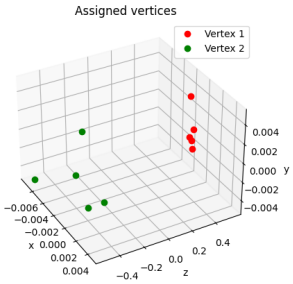
These Hamiltonians are known as *Ising Hamiltonians* [9], where the $\sigma_i^{(z)}$ is the third Pauli matrix for qbit number i. The characteristic function of the Max-Cut problem can be easily translated into an *Ising Hamiltonian* [10], operating on a set of qbits and whose measured values would be 0 or 1. This comes from the Since the solution of the Max-Cut problem involves finding the maximum of the characteristic function, the Hamiltonian can be defined with a minus sign, in order to convert the maximum into a minimum. Once a Max-Cut problem has been defined and the characteristic function has been translated into a Hamiltonian, the VQE algorithm can be used in order to find the set of qbit states that minimize it. This collection of states will be the solution to the Max-Cut problem.

$$H \equiv -F \Rightarrow \max F \equiv \min H \tag{5}$$

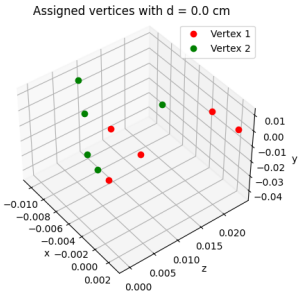
The vertex reconstruction problem with two vertices can be seen as a Max-Cut problem, in which the graph is composed by the 3-dimensional Point-Of-Closest-Approach of the tracks and the edges connecting every track contain a weight equal to the corresponding euclidean distance between the two. In this way, the characteristic function will have a maximum for the case in which the total distance between the two groups is maximal.

4 Results

The purpose of this chapter is to describe the results obtained by using the algorithm. There are two main focuses: measuring the accuracy of correctly assigning the tracks to the right vertex and understanding the dependency of the accuracy on the number of Hamiltonian



(a) Example of reconstructed track origins, with their respective vertices assigned. Vertices are generated in $d = \pm 0.5$ cm, thus are quite separated.



(b) Example of reconstructed track origins, with their respective vertices assigned. Vertices are generated in $d = 0.0$ cm, so exactly in the same spot.

Figure 2: Track to vertex assignment (in colours) for two different true distances measured in cm between the vertices.

evaluations.

4.1 Vertex generation

The data used to test the algorithm has been produced synthetically by a simplified track simulator considering a tracker of concentric layers located at increasing distances from the centre. Events have been designed to have a total of two vertices with 5 tracks each. Tracks have been smeared according to typical tracker resolution estimates and then reconstructed using a likelihood function. The true position of the vertex has been fixed to lie along the Z axis, in such a way that the two vertices are at positions d and -d, and therefore 2d is the distance between them. Several values of the distance have been taken into account. To study the dependence of the accuracy with the number of evaluations of the Hamiltonian this parameter has also been varied. For every distance and number of evaluations a total of 100 events have been generated in order to study the algorithm performance. An example of one of these vertices with distance parameter $d = 0.5$ cm can be seen in Figure 2a and another system with $d = 0.0$ cm may be seen in Figure 2b. It is noticeable that the distribution will be way easier to discriminate for the first configuration compared to the second one. There, each colour represents the vertex the point is coming from. This specific example is quite obvious at first sight, but it is also a realistic situation, with reasonable distances.

4.2 Algorithm accuracy metric: efficiency

As mentioned before, there is a probabilistic factor which influences the result obtained. Therefore, it is interesting to check the efficiency of the algorithm, in order to decide how many evaluations should give a reliable result. To measure this efficiency, a new term, that will be called ε is used. This ε is obtained as the ratio of tracks correctly associated to their vertex over the total number of tracks.

It is important to note the range of ε. Being a ratio, the maximum value it may take is 1, or, as it has been done in percentage, 100%. However, the minimum value is not 0. The theoretically lowest possible would be exactly 50%. As it is defined, the algorithm tries to group the tracks with respect to their origin vertex. However, it does not discriminate between either of the two, just focusing on the relationship of the tracks. Therefore, the information it provides, is whether a group of tracks is coming from the same vertex or not. This is easily understood with an example. Consider a set of vertices distributed as [0, 0, 0, 0, 0, 1, 1, 1, 1, 1], one vertex is labelled as '0' and the other one as '1'. The worst possible reconstruction would be [1, 0, 0, 0, 0, 1, 0, 0, 0, 0], where the correctly assigned vertices are the first four zeros and the second number one: $\varepsilon = 50\%$. If another track changed origin to "make it worse" it would just swap the relationship: [1, 1, 0, 0, 0, 1, 0, 0, 0, 0]. Now, the first vertex is labelled as '1' and the second one as '0'. Hence having $\varepsilon = 60\%$. This means that it cannot be wrong in more than half of the tracks, as otherwise the track grouping would just swap which of the vertices said tracks are assigned to.

Another interesting characteristic of the way efficiency is calculated is the value ε takes if the system is randomly reconstructed. A test was made, assigning the tracks randomly to either of the vertices for 100000 cases and computing the efficiency. Another interesting characteristic of the way efficiency is calculated is the mean value that ε takes if the system is randomly reconstructed. A test was made, assigning the tracks randomly to either of the vertices for 100000 cases and computing the efficiency. The mean value was $\bar{\varepsilon} = 68\%$.

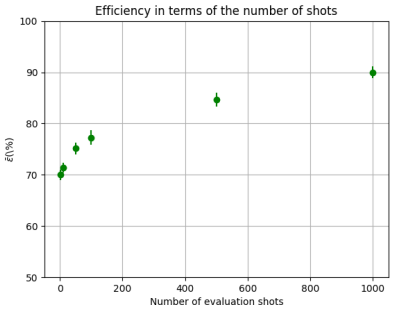
4.3 Efficiency dependence on the number of shots

To study the mentioned ε two vertices have been generated. First, the distance between them has been fixed and the number of evaluation shots has been variated, to look for the number that more or less makes the result reliable. Afterwards, this number has been set and the distance between vertices has been the one that varied.

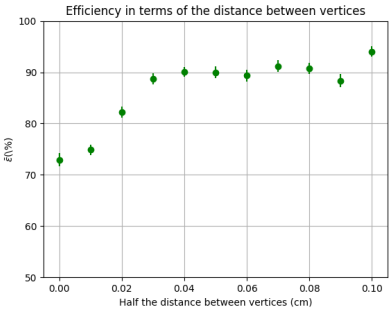
As it may be seen in Figure 3a, for a lower number of evaluation shots, the algorithm is not too reliable, with values close to the ones obtained with random assignments. Basically, with a low number of shots, it is almost randomly reconstructing the vertices. However, from 500 shots, the efficiency gets to a value of $\varepsilon = 84.68$ %, which is quite decent. It also reaches $\varepsilon = 90.00$ % with 1000 shots. The evaluation shots number ranges from 1 to 1000.

4.4 Efficiency dependence on the distance

It can be seen in Figure 3b that the efficiency of the algorithm grows considerably with distance for the reconstruction with 1000 shots per iteration. That is more than reasonable, as, the further the vertices are from each other, the easier it is to differentiate them. The distance from the vertices to $z = 0$ range from 0.0 to 0.1 cm. It is interesting to mention that, when both vertices are generated in the same spot, the efficiency comes really close to that of the randomly assigned vertices, which makes sense, as the vertices are basically undistinguishable. There is also a noticeable dip in the efficiency for the distance $dz = 0.09$ cm, which can be explained by a probabilistic indetermination. The drop in efficiency is not too significant, as it has $\varepsilon = 88.35$ % compared to the adjacent distances (90.73 and 94.04, respectively for 0.08 and 0.1 cm). Another surprising result is the fact that ε does not tend to 100%. This



(a) Epsilon values in term of the number of evaluations shots used in every step of the VQE with their error. The vertices are, respectively, in $z = -0.05$ and $z = 0.05$. Every step has been iterated 100 times, in order to obtain a useful mean value and standard deviation.



(b) Epsilon values in term of the distance between the vertices. The number of evaluations shots used in every step of the VQE is 1000, as that is a not too large number that gives somewhat reliable results. Every step has been iterated 100 times, in order to obtain a useful mean value and standard deviation. Note that, as the distance from the vertices to the origin is the same for the two generated vertices, the distance shown in the figure is actually the one from $z = 0$ to the vertex, which is half the distance between the vertices.

Figure 3: Plots for two different distributions of vertices

should theoretically happen, but there must be some issue causing a lack of convergence for large distances. This could probably be solved by raising the number of shots.

It is not strange to see that there are still some occasions when the system distribution obtained is not the real one (as can be seen in the example of Figure 4), specially for a lower value of shots.

5 Conclusions

The algorithm shows the capability to correctly reconstruct the vertices in most of the cases when they are separated enough. It requires a decent amount of evaluation shots, this value being a minimum of around 500, where, for a reasonable distance, the efficiency gets to about 85%. For an even more reliable result, which might be interesting, especially for the shorter separations, the number of shots may be raised up to 1000. Tested with 1000 evaluations per step, and with distances between 0.0 and 0.1 cm, the algorithm returns an efficiency of almost 90% with distances from 0.3 cm. It was expected to achieve an even higher efficiency, especially for the larger separations, however, there seems to be a saturation issue that makes it so that the efficiency does not get to values close to 100%. This might be solved by increasing the number of evaluation shots in the future.

In any case, this work shows the proof of concept that a quantum computing algorithm can be used to solve the problem of the vertex reconstruction opening a very interesting field where surely many parameters can be optimized to get better results.

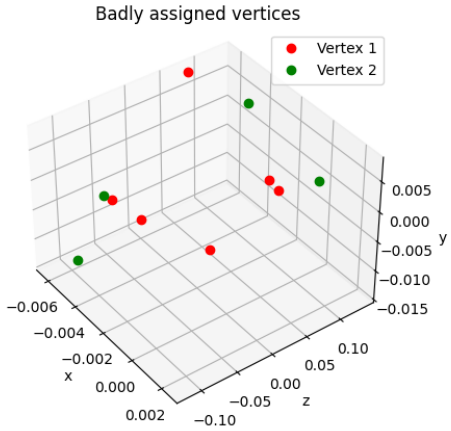


Figure 4: This is an example of a badly assigned set of vertices with $z = -0.1$cm and $z = 0.1$cm, having $\varepsilon = 60\%$, and being quite clear that the assignation is not the correct one.

References

[1] G. Benenti, G. Casati, G. Strini, *Principles of Quantum Computation and Information*, Vol. 1 (World Scientific, 2007)

[2] G. Benenti, G. Casati, G. Strini, *Principles of Quantum Computation and Information*, Vol. 2 (World Scientific, 2007)

[3] A. Peruzzo, J. McClean, P. Shadbolt, M.H. Yung, X.Q. Zhou, P.J. Love, A. Aspuru-Guzik, J.L. O'Brien, Nature Communications **5** (2014)

[4] F. Bloch, Nuclear introduction (1946)

[5] J.R. McClean, J. Romero, R. Babbush, A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, 023023.Âădoi:10.1088/1367-2630/18/2/023023 (2016)

[6] M.J. Powell, in *Advances in optimization and numerical analysis* (Springer, 1994), pp. 51–67

[7] S.P. Lloyd, *Least squares quantization in pcm* (1957)

[8] *Deterministic Annealing for clustering, compression, classification, regression and related optimization problems*, Vol. 86 (Proceedings IEE, 1998)

[9] L.W. McKeehan, Phys. Rev. **26**, 274 (1925)

[10] Y. Haribara, S. Utsunomiya, Y. Yamamoto, *A Coherent Ising Machine for MAX-CUT Problems: Performance Evaluation against Semidefinite Programming and Simulated Annealing* (Springer Japan, Tokyo, 2016), pp. 251–262, ISBN 978-4-431-55756-2, https://doi.org/10.1007/978-4-431-55756-2_12