Graphs on Qubits: Demonstrating Three Graph Algorithms on Quantum Computers

Lauri Vuorenkoski
University of Helsinki
lauri.vuorenkoski@gmail.com

Valter Uotila
University of Helsinki
valter.uotila@helsinki.fi

ABSTRACT

Graph algorithms are at the center of database optimization and data management, and countless optimization algorithms can be expressed as graph algorithms. Additionally, graph algorithms form the foundational mechanism to query graph databases. In this work, we demonstrate how three central graph problems, all-pairs shortest path, graph isomorphism, and community detection, can be solved using quantum annealers. The problem formulation for the all-pairs shortest path problem is new. The work is implemented in a demonstration system, which shows that the selected graph algorithms have natural quantum computational formulations and that running small but realistic workloads on quantum annealers is feasible.

VLDB Workshop Reference Format:
Lauri Vuorenkoski and Valter Uotila. Graphs on Qubits: Demonstrating Three Graph Algorithms on Quantum Computers. VLDB 2024 Workshop: The Second International Workshop on Quantum Data Science and Management (QDSM'24).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/vuorenkoski/graphs_on_qubits.

1 INTRODUCTION

Quantum computing for databases and data management is an emerging research field that has seen substantial growth in recent years [35, 46]. The field aims to address the need for more sophisticated optimization methods, which are crucial as the volume and complexity of data continue to grow at an increasing rate. The key vision in the field is that the optimization of databases could happen partly on quantum computers in the future.

Most of the previous research has focused on the optimization of relational databases utilizing various quadratic unconstrained binary optimization formulations [4, 11, 15, 17, 25, 33, 34, 36, 37, 44, 45, 49, 53]. The second most common quantum computational approach is to tackle database problems with quantum machine learning [18, 19, 47, 51, 52]. Although many optimization problems in relational databases are fundamentally graph problems (e.g., join order selection), the full power of graph algorithms in the field is not yet well-studied. To start a more systematic study and benchmarking of the existing graph algorithms in quantum computing, we have developed a demonstration system that allows users to study three central graph algorithms: all-pairs shortest path, graph isomorphism, and community detection.

The primary motivation behind this demonstration system is the necessity of understanding fundamental graph algorithms, which have a crucial role in many real-life applications. We also likely need to develop specific quantum algorithms instead of rewriting classical algorithms in the quantum equivalent format. For example, this type of algorithm development was demonstrated in [48]. It is also important to identify which types of graph algorithms have characteristics that would benefit from quantum computing. For example, a clear example is the shortest path finding algorithms, which have polynomial-time classical solutions but relatively expensive quadratic unconstrained binary optimization formulations [21]. On the other hand, quantum walks on graphs have proved to show speedups [24, 41]. This work demonstrates these types of differences between the studied algorithms.

First, we present a brief background to quadratic binary optimization on quantum computers. Then, we define the three graph algorithms, all-pairs shortest path, graph isomorphism, and community detection, as binary optimization problems. Finally, we will present the demonstration system and show how it can be utilized to obtain results. This work was initially developed for the master's thesis [50].

1.1 Binary optimization on quantum computers

For a comprehensive introduction to quantum computing, we guide the reader to [29]. Quantum machine learning for database optimization was covered in [51]. Ising problems, equivalent to unconstrained binary optimization problems, and a large number of NP-hard problems using binary models are introduced in [23]. Also, [34] contains a good introduction to Quadratic Unconstrained Binary Optimization (QUBO) formalism and discusses the capabilities of the current quantum annealers.

Next, we describe the key, high-level idea behind the optimization processes on quantum hardware. The optimization problems can be represented as Hamiltonians, which are essentially matrices that describe the so-called energy levels of the corresponding quantum system. Our focus is on computing the smallest eigenstates and eigenvectors of these matrices as they correspond to the ground energy states for Hamiltonians. If the optimization problem is encoded correctly, the low-energy states also represent solutions to the original real-life optimization problem.

Quantum computing can be divided into at least two paradigms: circuit-based quantum computing and adiabatic quantum computing. Quantum annealers, to which this paper is focused, are based on adiabatic quantum computing. The high-level idea of adiabatic quantum computing is that the computation starts from an initial

Proceedings of the VLDB Endowment. ISSN 2150-8097.

Hamiltonian with a simple and known ground state and proceeds to a final Hamiltonian whose ground state encodes the solution to the computational problem [3]. If the transformation from the simple Hamiltonian to the problem Hamiltonian is sufficiently slow, the adiabatic theorem ensures that the system remains in the lowest-energy state throughout the entire process.

D-Wave is the leading developer of quantum annealers. The first D-Wave quantum computer was built in 2011 [31]. In 2024, there are three versions of the D-Wave annealers in open use [1]: Advantage system 4.1 (5627 qubits and 40279 couplers), Advantage system 5.4 (5614 qubits and 40050 couplers), and Advantage system 6.3 (5614 qubits and 40105 couplers). D-Wave provides both quantum solvers and hybrid solvers. Quantum solvers use quantum annealers directly, whereas hybrid solvers use both classical and quantum resources to solve optimization problems. On the software level, D-Wave provides Ocean, which is a Python framework through which quantum annealers can be accessed.

Next, we describe our theoretical formalism, which enables us to express a wide variety of optimization problems. Let $x = {}^{1}x_1, \ldots, x_n{}^{o} \; 2 \; \mathrm{f}0, 1\mathrm{g}^n$ a binary vector. A Quadratic Unconstrained Binary Optimisation (QUBO) problem is a combinatorial optimization problem where the goal is to minimize the objective function $f\colon \; \mathrm{f}0, 1\mathrm{g}^n \;\; ! \;\; \mathbb{R}$ defined by

$$f^{1}x^{o} = \tilde{\sum_{i=1}^{m}} Q_{i,i}x_i \; , \; \tilde{\sum_{i<j}} Q_{i,j}x_ix_j.$$

The diagonal terms $Q_{i,i}$ are the linear coefficients, and the off-diagonal terms $Q_{i,j}$ are the quadratic coefficients. Solving QUBO is an NP-hard problem [23] that has a wide variety of applications in different fields. There are many classical algorithms to solve QUBO problems [20], and they are the standard formalism to express problems for quantum annealers.

Quantum annealers are restricted by topologies, which define the qubit connections in the device. While mapping QUBO problems to annealing topologies is an important problem, our demonstration will rely on the standard "minor miner" algorithm defined in the Ocean software. Since the mapping is an NP-hard problem [22] itself, in some cases, it is useful to identify faster heuristics to perform the mapping [44, 45].

2 ALGORITHMS

We have selected three distinct types of graph problems:

(1) All pairs shortest path [39]
(2) Graph isomorphism [16]
(3) Community detection [10]

These problems have been chosen to represent different computational complexity classes, which are visualized in Figure 1. While intuitively solving P-hard problems should be the easiest, the problem encoding on quantum hardware creates a certain overhead that seems problem-dependent. Interestingly, the NP-hard community detection problem seemed to have the best scalability on quantum computers among the three selected graph algorithms.

2.1 All-pairs shortest path

The shortest path-finding problem is one of the most well-known computer science problems. Since the Dijkstra algorithm solves it

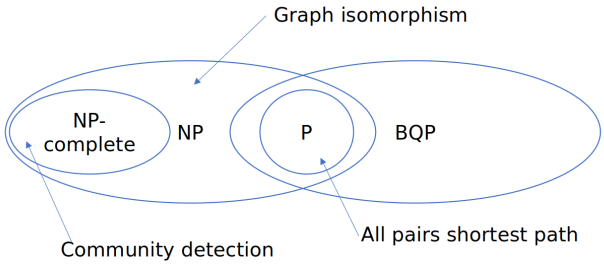

Figure 1: Relation of the complexity classes and selected algorithms.

in polynomial time, the quantum computational formulation for the path-finding algorithm works as an example of quantum algorithm design rather than an attempt to find a formulation that would beat the classical algorithms.

There exist three varying QUBO formulations and an adiabatic algorithm for the single-pair shortest path algorithm [21, 30]. In this subsection, we are developing a new type of quantum annealing algorithm where we are not only interested in finding the single shortest path but the shortest paths between multiple pairs, likely all of them. Each minimal solution corresponds to the shortest path between certain vertices in a graph. Assuming that we will obtain a sufficient number of samples from the quantum annealer, we would solve the all-pairs shortest path algorithm with a single QUBO. This problem is still not hard since the Floyd-Warshall algorithm [8] solves the problem in polynomial time. However, the Floyd-Warshall algorithm produces only the sum of the path weights when our algorithm also produces the path.

Binary variables. Let $G = {}^{1}V, E^{o}$ be a weighted and directed graph with weights $w_{i,j} > 0$ for each $e_{i,j} \; 2 \; E$. We define $2\mathrm{j}V\mathrm{j} \; , \; \mathrm{j}E\mathrm{j}$ binary variables representing start node, target node, and edges. The first $\mathrm{j}V\mathrm{j}$ binary variables, corresponding to the starting vertices of shortest paths, are x_i^s for $i \; 2 \; V$ where the superscript s refers to "start". These variables are one-hot encoding of the starting vertex s for each path. If $x_i^s = 1$, then the vertex i is the starting vertex of the corresponding shortest path. Next, variables x_i^t for $i \; 2 \; V$ are similarly one-hot encoding of the target vertices. Last, we define variables x_{ij}^e which are one-hot encodings of edges $e_{ij} \; 2 \; E$. The superscript clarifies that the variable refers to edges. If $x_{ij}^e = 1$, then e_{ij} is part of the shortest path.

Next, we encode multiple constraints. It is important to remember that we are not encoding a problem where we want to find a single minimal solution but a problem where the first low-energy solutions represent certain shortest paths in a graph. In the following constraints, we encode the validity of a solution and choose the coefficient of $p = \; {}^{1}_{1}{}_{i,j}{}^{o}{}_{2E} \, w_{i,j} \; , \; 1$. The coefficient is adjusted so that all solutions with energy levels below zero are correct paths, but not necessarily shortest paths. The minimum value of the cost constraint is the sum of weights. Even a small error in path formation should counterbalance this.

Uniqueness of starting vertices. Every path must have a single starting vertex s. To encode the constraint, we use the standard

"select one variable to be true" constraint

$$H_1 = p\ 1 \Big(\sum_{i \in V} x_i^s \Big)^2 .$$

Next, we encode that there must be an edge starting from the starting vertex s and no edge terminating to s. This constraint returns p if the starting vertex and edge from the vertex match. The constraint penalizes cases if the starting vertex and edge terminating in that vertex match. Thus, the next constraint becomes

$$H_2 = p \sum_{e_{ij} \in E} x_i^s x_{ij}^e , p \sum_{e_{ij} \in E} x_j^s x_{ij}^e .$$

Uniqueness of target vertices. Similarly to the previous uniqueness constraint, there has to be a unique target vertex t for every path. The constraint penalizes every such pair if more than one target vertex is selected. Hence, the full constraint is

$$H_3 = p\ 1 \Big(\sum_{i \in V} x_i^t \Big)^2 .$$

The next constraint ensures that there must be an edge terminating to t and no edge starting from t. The constraint returns p if the target vertex and edge terminating to that vertex match. The constraint returns p if the target vertex and edge starting from that vertex match:

$$H_4 = p \sum_{e_{ij} \in E} x_i^t x_{ij}^e , p \sum_{e_{ij} \in E} x_j^t x_{ij}^e .$$

Finally, the starting vertex s and the target vertex t must differ. In this case, the constraint penalizes with p if both x_i^s and x_i^t are selected.

$$H_5 = p \sum_{v_i \in V} x_i^s x_i^t$$

Constraints on edges. The next constraint encodes that two edges should not start or terminate at the same vertex. Forcing this constraint ensures that the path does not contain cycles:

$$H_6 = p \sum_{e_{ij} \in E} \Big(\sum_{\{e_{il} \in E} x_{ij}^e x_{il}^e , \sum_{e_{kj} \in E} x_{ij}^e x_{kj}^e \Big)^2 .$$

The next constraint encodes the property that every path has to be connected:

$$H_7 = p \sum_{e_{ij} \in E} \Big(-x_{ij}^e \sum_{e_{jl} \in E} x_{ij}^e x_{jl}^e \Big)^2 .$$

In other words, if we include edge $\langle i, j \rangle$, then we should have exactly one edge $\langle j, l \rangle$ which is connected to edge $\langle i, j \rangle$.

Cost constraint. Paths having lower weights should be prioritized. This final constraint returns w_{ij} for each edge selected to the path:

$$H_8 = \sum_{e_{ij} \in E} w_{ij} x_{ij}^e .$$

2.2 Graph isomorphism

The problem of determining if two graphs are isomorphic is not known to be NP-complete, although it is a hard problem in practice [16]. Graph isomorphism is particularly interesting because, similar to the integer factoring problem solved by Shor's algorithm in polynomial time [40], it is not known to be NP-complete.

Formally, a graph isomorphism problem is to find a bijective mapping $\pi : V_1 \to V_2$ for $G_1 = \langle V_1, E_1 \rangle$ and $G_2 = \langle V_2, E_2 \rangle$ so that whenever $\langle v_1, w_1 \rangle \in E_1$ is an edge in G_1, then $\langle \pi\langle v_1 \rangle, \pi\langle w_1 \rangle \rangle \in E_2$ is an edge in G_2. The QUBO formulation for the graph isomorphism problem was presented in [7, 23]. The other quantum computing formulations are the quantum walk-based algorithm [42], and the adiabatic algorithm [12]. The comparison of quantum algorithms for graph isomorphism problems is [9]. In this formulation, we use a coefficient $p = |V|$ in the bijectivity constraint. The bijectivity constraint will not be violated with this approach.

Binary variables. The binary variables are x_{v_1,v_2} for each $v_1 \in V_1$ and $v_2 \in V_2$. If $x_{v_1,v_2} = 1$, we set $\pi\langle v_1 \rangle = v_2$.

Bijectivity. First, we construct a constraint for bijectivity:

$$H_1 = p \sum_{v_1 \in V_1} \Big(-1 + \sum_{v_2 \in V_2} x_{v_1,v_2} \Big)^2 , p \sum_{v_2 \in V_2} \Big(-1 + \sum_{v_1 \in V_1} x_{v_1,v_2} \Big)^2 .$$

The first term forces the constraint that for every vertex $v_1 \in V_1$ in the domain graph, we choose exactly one vertex $v_2 \in V_2$ in the target graph. Since we want to construct a bijection, the second term encodes the same mapping in the other direction.

Mapping respects edges. The second constraint encodes the property that the graph isomorphism must respect the edges:

$$H_2 = \sum_{\langle v_1, v_2 \rangle \in E_1} \sum_{\langle w_1, w_2 \rangle \in E_2} x_{v_1,w_1} x_{v_2,w_2} + \sum_{\langle v_1, v_2 \rangle \in E_1} \sum_{\langle w_1, w_2 \rangle \in E_2} x_{v_1,w_2} x_{v_2,w_1}$$

Considering the first term, if there is an edge $\langle v_1, v_2 \rangle \in E_1$ on the domain side, then we should have an edge $\langle w_1, w_2 \rangle \in E_2$ on the target side so that $\pi\langle v_1 \rangle = w_1$ and $\pi\langle v_2 \rangle = w_2$. If we do, the energy level is lowered by -1. The second term encodes the same constraint in the other direction. This is needed as the graphs are undirected.

The final QUBO is the sum of the previous objectives. In this case, the QUBO has a minimum of $-|E|$ if and only if the graphs are isomorphic [23].

2.3 Community detection

The community detection algorithm partitions an undirected graph into communities so that the number of edges in the same community is maximized and the number of edges between the communities is minimized [10, 43]. The algorithm is NP-complete [5]. Community detection has many applications, for example, in computer science, social sciences, and biology [10, 14]. Unlike graph isomorphism or shortest paths problems, the community detection problem does not have an exact solution for partitioning a graph into communities, as there is no precise definition of a community. Community detection has been relatively widely studied from a quantum computing perspective [2, 6, 13, 26, 32, 38].

Several algorithms are proposed to solve community detection problem [10, 28]. One approach is constructing a function that describes the quality of the chosen partitioning. Then, the goal is to express this quality function as a QUBO problem and maximize its value. The QUBO formulation used in this work is based on [26]. It relies on the modularity approach [27], where the task is to maximize the function

$$Q = \frac{1}{2m} \sum_{ij} \left(w_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j),$$

where k_i is the sum of weights of edges that connect to vertex i, c_i is the community of vertex i, w_{ij} is the weight between vertices i and j, $m = \frac{1}{2} \sum_{ij} w_{ij}$ is the total sum of weights in the graph and characteristic function $\delta(i, j)$ is 1 if $i = j$ and 0 otherwise. The function describes the modularity of a partition of a graph. A value of 0 means that the partition is random. The higher the value, the better the modularity is. Maximum value is 1. Values around 0.3 or more usually indicate good partitioning [27].

In this formulation, we use a coefficient $p = 0.1$ in the first constraint. The algorithm's performance would be much poorer without this, probably due to the significant difference in magnitude between the values in the first and second constraints.

Binary variables. For the QUBO encoding, we define $|V||C|$ many binary variables as $x_{vc} \in \{0, 1\}$, where $v \in V$ and $c \in C$ indicating if vertex v belongs to the community c. We construct the objective function from the following constraints.

Vertex must belong to exactly one community. This constraint is the standard "select one variable from a set of variables". The objective function reaches its minimum when we have selected exactly one community for each vertex:

$$H_1 = p \sum_{c \in C} \left(1 - \sum_{v \in V} x_{vc} \right)^2.$$

Minimise modularity. The function returns modularity value $k_i k_j / 2m - w_{ij}$ for each pair of vertices i and j if they belong to the same community. Since we want to maximize the modularity and deal with minimization problems, the modularity for each pair is multiplied by -1. Thus, this constraint becomes:

$$H_2 = \frac{1}{2m} \sum_{c \in C} \sum_{i \in V} \sum_{j \in V} \left(\frac{k_i k_j}{2m} - w_{ij} \right) x_{ic} x_{jc}.$$

3 DEMONSTRATION SYSTEM AND RESULTS

We have implemented the previous three graph algorithms in a demonstration system. The system has a user-friendly front end, which allows users to explore the three graph problems with many different types of preloaded graphs. User can also upload their own graphs. The problems can be solved with various D-Wave quantum computers and simulators.

The outputs are visualized in multiple ways, which are shown in Figure 2. The demo system visualizes the underlying graph, provides an overview of the QUBO weights with an illustrative heatmap, prints out energy levels and their occurrences, and represents other relevant information about the measurements from the quantum annealer.

Our goal was not to develop a scientifically rigorous benchmark system but to demonstrate how the selected graph algorithms can

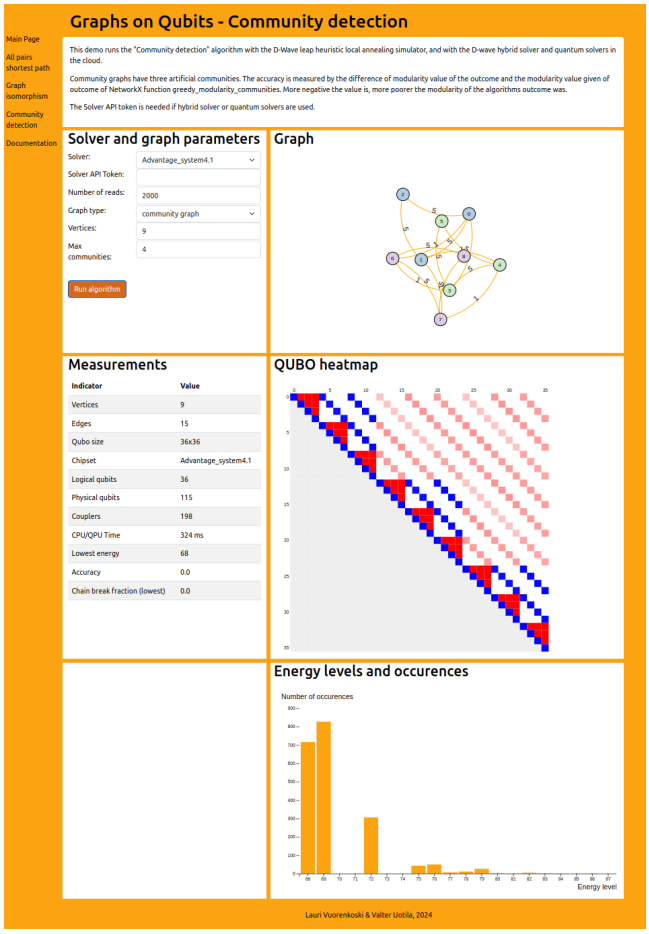

Figure 2: Screenshot of the demo system after executing community detection algorithm

be designed using QUBO formulation, implemented with D-Wave Ocean software, and integrated with the demo system so that their execution would be easy for non-specialists. We plan to extend this demo into a realistic benchmark in future work. Using the demo system, we can obtain some results, which we will present and discuss in this section. More detailed results can be found in [50]. We want to emphasize that it is often a complex task to find correct weights between QUBO constraints, which affects the quality of the results. Thus, there might be room for further improvement regarding these results.

The results presented here include only a few of the graph types available in the demonstration system. A random graph is constructed by NetworkX `gnp_random_graph` function with probability for edge creation 0.3 and seed 42. The all-pairs shortest path and the community detection algorithms use graphs having weights. In these cases, random weight $1 \ldots 10$ is placed on all edges (except the community graph). In the community graph, vertices are divided into three approximately equal-sized groups. Every group is itself a complete subgraph in which each edge has a weight of 5. Two

vertices from each group are connected to two vertices in all other groups with edges having weights 1.

In practice, algorithms are executed on a quantum annealer several times to achieve the correct lowest energy level. In the demonstration system, this parameter can be set in the "number of reads" parameter. This defines how many samples an annealer produces as output. By default, we set it to 2000. Increasing this parameter would increase the running time but would give better accuracy.

3.1 All-pairs shortest path algorithm

Considering the all-pairs shortest path algorithm, we demonstrate the algorithm with the graph in Figure 3. After constructing the QUBO problem for this graph and executing it on a quantum annealer, the service returns a summary of samples from the annealer. Some of the samples are shown in Figure 4.

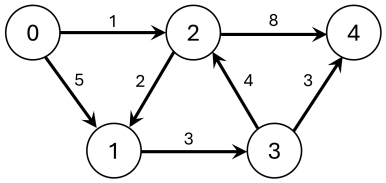

Figure 3: Graph example used in all pairs shortest path problem

```
   0-1 0-2 1-3 2-1 2-4 3-2 3-4 s0 s1 s2 s3 s4 t0 t1 t2 t3 t4 energy num_oc.
0    0   1   0   0   0   0   0  1  0  0  0  0  0  1  0  0  0  -26.0      61
1    0   0   0   1   0   0   0  0  0  1  0  0  0  1  0  0  0  -25.0     279
2    0   1   0   1   0   0   0  1  0  0  0  0  0  1  0  0  0  -24.0     139
...
8    0   1   1   1   0   0   0  1  0  0  0  0  0  0  0  1  0  -21.0      82
...
['BINARY', 27 rows, 1890 samples, 17 variables]
```

Figure 4: Example of results from quantum solver in all pairs shortest path.

The idea of how to interpret these samples is as follows: If we choose to interpret the sample on row number 2, we see that the variables 0 2, 2 1, $s0$, and $t1$ are set to 1. This result means that we have found the shortest path from starting vertex 0 ($s0$) to target vertex 1 ($t1$), which consists of edges ^{1}0, 2^0 and ^{1}2, 1^0. This is the correct solution since this path has weight 3 in Figure 3, while any other path is more expensive. Similarly, every row in the solution having a negative energy level describes one of the correct paths in the graph. The shortest paths have the smallest energy level.

Now, the same algorithm has been benchmarked with various other tree, wheel and random graphs. The results are in Figure 5. The results are obtained from a simulator and D-Wave's quantum annealer without using D-Wave's hybrid features. The accuracy is compared to the classical all-pairs shortest path algorithm.

Both the local simulator and the quantum solver produced the correct output for graphs with 5 vertices. The results were similar for graphs with 6 vertices, except that the quantum solver reached only an accuracy of 93 % with the random graph.

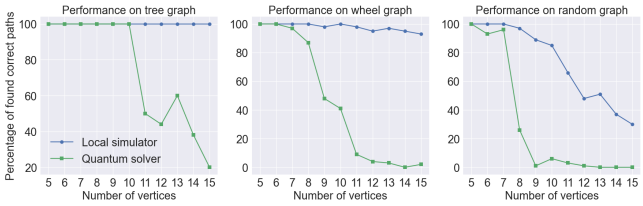

Figure 5: Results from all-pairs shortest path algorithm

Performance was significantly dropped when the algorithm was applied to graphs with more vertices. The algorithm performed well on the local simulator but struggled on the quantum solver. Results were significantly better on the tree graphs than on the wheel graphs and especially on the random graphs. The tree graphs are more sparse than wheel graphs, and wheel graphs are more sparse than random graphs. When applying this algorithm, more dense graphs mean that more couplers (quadratic coefficients) are needed, which mainly explains the differences. Current quantum annealers are constructed so that there is a limited amount of physical couplers. If there is a need for more couplers, one logical qubit must be represented by two physical qubits that are coupled together. This makes the actual qubit-coupler construction more complex, which then brings more noise to the algorithm.

In general, accuracy drops rather steeply when the number of vertices is increased. This is partly explained by the limitations of a real quantum computer but partly due to the nature of the algorithm. To identify all possible correct paths, we need polynomially more correct samples. The more vertices and edges a graph has, the more probable it is that all of these paths do not fit in the sample set (which in these benchmarks was set to 2000).

3.2 Graph isomorphism

The results of the graph isomorphism algorithm are presented in Figure 6. Performance was measured using pairs of isomorphic graphs. The second graph was created from the first graph by randomly permuting the vertices. Only isomorphic graphs were used because, in practice, failure of the algorithm can only be observed with isomorphic graphs, as non-isomorphic graphs always produce energy levels greater than jEj (which is the correct energy level of isomorphism). The performance was measured by the difference between the energy level of the outcome and the correct energy level.

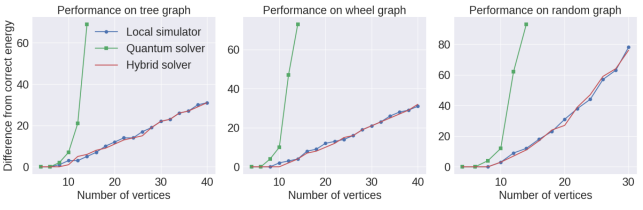

Figure 6: Results from graph isomorphism algorithm

The local simulated annealing solver, the hybrid solver, and the quantum solver performed very well, with the small-sized graphs

having less than 8 vertices. When the number of vertices was increased from that level, all three solvers started to struggle. Solvers found a rather low energy level, but not the lowest one, which would indicate correct output. This gap grew larger when more vertices were added to the graphs.

The hybrid solver and local solver performed similarly, but there was a significant difference in running time. The hybrid solver took a rather constant 3 seconds to run when the local simulator could take more than 30 seconds in the largest graphs.

Again, random graphs were the hardest problem for the algorithm. The same rule applies to this algorithm as to the all-pairs shortest path algorithm: the more edges the graphs have, the more couplers (quadratic coefficients) are needed.

3.3 Community detection

The results of the community detection algorithm are presented in Figure 7. Although community detection is an NP-complete problem, the results are one of the most promising for this algorithm. The main indicator for performance was the accuracy measured by the difference of modularity value of the outcome (lowest energy level found) and the modularity value calculated from the outcome of classical NetworkX function `greedy_modularity_communities`.

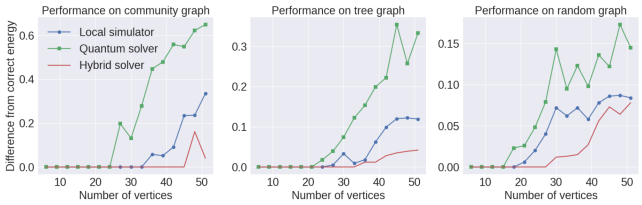

Figure 7: Results from community detection algorithm

The most important graph type for benchmarking this algorithm was the community graph, which has three clear communities, as described earlier. The algorithm was also benchmarked with tree graphs and random graphs. The tree graphs can have rather clear communities depending on random weights, while the random graphs we used here do not have any clear communities.

The performance started to drop much slower with this algorithm than with two other algorithms when the number of vertices was increased. When using community and tree graphs, the algorithm produced correct results in graphs having 20 vertices. When the number of vertices increased from that, the hybrid solver had the best performance, even better than the local simulator. The quantum solver had again the most difficulties in reaching the correct energy level.

One explanation for better performance in this algorithm could be that, for example, the tree graph having 20 vertices needs 400 qubits and 8322 couplers in the graph isomorphism problem but only 60 qubits and 630 couplers when solving the community detection problem. However, the need for qubits and couplers in this algorithm is quite the same as when solving the all-pairs shortest path problem (the tree graph having 20 vertices needs 59 qubits and 502 couplers). This is explained by the fact that the all-pairs shortest path algorithm has other performance constraints explained earlier, mainly the need for many samples of varying negative energy levels.

4 CONCLUSION AND FUTURE WORK

In this paper, we have developed a demonstration system that implements three central graph algorithms: all-pairs shortest path, graph isomorphism, and community detection. We invite users to download the demo on GitHub and test it. Using the system, we obtained initial results that describe the scalability of the quantum algorithms and systems compared to the classical solvers. The system and the results also demonstrate differences between the algorithms at a very concrete level.

When constructing and testing algorithms, we found that fine-tuning the coefficients for the individual constraints is important. Firstly, it is important that the coefficients are such that the algorithm works mathematically correctly (all-pairs shortest path, graph isomorphism). However, it can also be important in improving the performance of algorithms in current noisy quantum computers (community detection).

We plan to collect a more systematic literature survey and a comprehensive implementation of graph algorithms on quantum computers in the future. As we argued in the beginning, the importance of graph algorithms is evident. Often, real-life solutions require small modifications or additional constraints that might be difficult to encode using the original formulations but these modifications might be natural to include using QUBOs. The flexibility and expressiveness of the QUBO formalism have been demonstrated in many applications, and we are planning to improve these formulations further in database research.

REFERENCES

[1] [n.d.]. The Advantage Quantum Computer | D-Wave. https://www.dwavesys.com/solutions-and-products/systems/

[2] Sana Akbar and Sri Khetwat Saritha. [n.d.]. Towards quantum computing based community detection. 38 ([n. d.]), 100313. https://doi.org/10.1016/j.cosrev.2020.100313

[3] Tameem Albash and Daniel A. Lidar. 2018. Adiabatic Quantum Computing. *Reviews of Modern Physics* 90, 1 (Jan. 2018), 015002.

[4] Tim Bittner and Sven Groppe. 2020. Avoiding blocking by scheduling transactions using quantum annealing. In *Proceedings of the 24th Symposium on International Database Engineering & Applications*. 1–10.

[5] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. [n.d.]. On Modularity Clustering. 20, 2 ([n. d.]), 172–188. https://doi.org/10.1109/TKDE.2007.190689 Place: New York, NY Publisher: IEEE.

[6] Chris Cade, Marten Folkertsma, Ido Niesen, and Jordi Weggemans. [n.d.]. Quantum Algorithms for Community Detection and their Empirical Run-times. ([n. d.]). https://doi.org/10.48550/arXiv.2203.06208

[7] Cristian Calude, Michael Dinneen, and Richard Hua. [n.d.]. QUBO formulations for the graph isomorphism problem and related problems. 701 ([n. d.]). https://doi.org/10.1016/j.tcs.2017.04.016

[8] Robert W. Floyd. [n.d.]. Algorithm 97: Shortest Path. 5, 6 ([n. d.]), 345. https://doi.org/10.1145/367766.368168 Place: New York, NY, USA Publisher: Association for Computing Machinery.

[9] Pasquale Foggia, Carlo Sansone, Mario Vento, and others. [n.d.]. A performance comparison of five algorithms for graph isomorphism. In *Proceedings of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition* (2001). Citeseer, 188–199. https://mivia.unisa.it/wp-content/uploads/2013/05/foggia01.pdf

[10] Santo Fortunato. [n.d.]. Community detection in graphs. 486, 3 ([n. d.]), 75–174. https://doi.org/10.1016/j.physrep.2009.11.002

[11] Kristin Fritsch and Stefanie Scherzinger. 2023. Solving Hard Variants of Database Schema Matching on Quantum Computers. *Proceedings of the VLDB Endowment* 16, 12 (Aug. 2023), 3990–3993. https://doi.org/10.14778/3611540.3611603

[12] Frank Gaitan and Lane Clark. [n.d.]. Graph isomorphism and adiabatic quantum computing. 89, 2 ([n. d.]), 022342. https://doi.org/10.1103/PhysRevA.89.022342

[13] Felix G. Gemeinhardt, Robert Wille, and Manuel Wimmer. [n.d.]. Quantum k-community detection: algorithm proposals and cross-architectural evaluation. 20, 9 ([n. d.]), 302. https://doi.org/10.1007/s11128-021-03239-1

[14] M. Girvan and M. E. J. Newman. [n.d.]. Community Structure in Social and Biological Networks. 99, 12 ([n. d.]), 7821–7826. Place: United States Publisher: National Academy of Sciences.

[15] Martin Gogeissl, Hila Safi, and Wolfgang Mauerer. 2024. Quantum Data Encoding Patterns and their Consequences. In *Proceedings of the 1st Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications (Q-Data '24)*. Association for Computing Machinery, New York, NY, USA, 27–37. https://doi.org/10.1145/3665225.3665446

[16] Martin Grohe and Pascal Schweitzer. [n.d.]. The graph isomorphism problem. 63, 11 ([n. d.]), 128–134. https://doi.org/10.1145/3372123

[17] Sven Groppe, Jinghua Groppe, Umut Çalıkyılmaz, Tobias Winker, and Le Gruenwald. 2022. Quantum Data Management and Quantum Machine Learning for Data Management: State-of-the-Art and Open Challenges. In *Proceedings of the EAI ICISML conference*.

[18] Le Gruenwald, Tobias Winker, Umut Çalıkyılmaz, Jinghua Groppe, and Sven Groppe. 2023. Index Tuning with Machine Learning on Quantum Computers for Large-Scale Database Applications. In *Joint Proceedings of Workshops at the 49th International Conference on Very Large Data Bases (VLDB 2023) - International Workshop on Quantum Data Science and Management (QDSM'23), Vancouver, Canada*. https://ceur-ws.org/Vol-3462/QDSM5.pdf

[19] Florian Kittelmann, Pavel Sulimov, and Kurt Stockinger. 2024. QardEst: Using Quantum Machine Learning for Cardinality Estimation of Join Queries. In *Proceedings of the 1st Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications (Q-Data '24)*. Association for Computing Machinery, New York, NY, USA, 2–13. https://doi.org/10.1145/3665225.3665444

[20] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. [n.d.]. The unconstrained binary quadratic programming problem: a survey. 28, 1 ([n. d.]), 58–81. https://doi.org/10.1007/s10878-014-9734-0

[21] T Krauss and J McCollum. [n.d.]. Solving the Network Shortest Path Problem on a Quantum Annealer. 1 ([n. d.]), 1–12. https://doi.org/10.1109/TQE.2020.3021921

[22] Elisabeth Lobe and Annette Lutz. [n.d.]. Minor Embedding in Broken Chimera and Pegasus Graphs is NP-complete. ([n. d.]). https://doi.org/10.48550/arXiv.2110.08325

[23] Andrew Lucas. [n.d.]. Ising formulations of many NP problems. 2 ([n. d.]), 5. https://doi.org/10.3389/fphy.2014.00005

[24] Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. 2011. Search via Quantum Walk. *SIAM J. Comput.* 40, 1 (Jan. 2011), 142–164. https://doi.org/10.1137/090745854

[25] Nitin Nayak, Jan Rehfeld, Tobias Winker, Benjamin Warnke, Umut Çalıkyılmaz, and Sven Groppe. 2023. Constructing Optimal Bushy Join Trees by Solving QUBO Problems on Quantum Hardware and Simulators. In *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments (BiDEDE), Seattle, WA, USA*. https://doi.org/10.1145/3579142.3594298

[26] Christian F A Negre, Hayato Ushijima-Mwesigwa, and Susan M Mniszewski. [n.d.]. Detecting multiple communities using quantum annealing on the D-Wave system. 15, 2 ([n. d.]), e0227538–e0227538. https://doi.org/10.1371/journal.pone.0227538

[27] M. E. J. Newman. [n.d.]. Analysis of weighted networks. 70, 5 ([n. d.]), 056131. https://doi.org/10.1103/PhysRevE.70.056131

[28] M E J Newman and M Girvan. [n.d.]. Finding and evaluating community structure in networks. 69, 2 ([n. d.]), 026113–026113. https://doi.org/10.1103/PhysRevE.69.026113

[29] Michael A. Nielsen and Isaac L. Chuang. 2010. *Quantum computation and quantum information* (10th anniversary ed ed.). Cambridge University Press, Cambridge; New York.

[30] Venkat Padmasola and Rupak Chatterjee. [n.d.]. Optimization on Large Interconnected Graphs and Networks Using Adiabatic Quantum Computation. ([n. d.]). https://doi.org/10.48550/arxiv.2202.02774 Place: Ithaca Publisher: Cornell University Library, arXiv.org.

[31] Atanu Rajak, Sei Suzuki, Amit Dutta, and Bikas K. Chakrabarti. [n.d.]. Quantum annealing: an overview. 381, 2241 ([n. d.]), 20210417. https://doi.org/10.1098/rsta.2021.0417

[32] Hannu Reittu, Ville Kotovirta, Lasse Leskelä, Hannu Rummukainen, and Tomi Räty. [n.d.]. Towards analyzing large graphs with quantum annealing and quantum gate computers. ([n. d.]). https://doi.org/10.48550/arXiv.2006.16702

[33] Pranshi Saxena, Ibrahim Sabek, and Federico Spedalieri. 2024. Constrained Quadratic Model for Optimizing Join Orders. In *Proceedings of the 1st Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications (Q-Data '24)*. Association for Computing Machinery, New York, NY, USA, 38–44. https://doi.org/10.1145/3665225.3665447

[34] Manuel Schönberger, Stefanie Scherzinger, and Wolfgang Mauerer. 2023. Ready to Leap (by Co-Design)? Join Order Optimisation on Quantum Hardware. *Proc. ACM Manag. Data* 1, 1, Article 92, 27 pages.

[35] Manuel Schönberger. 2022. Applicability of Quantum Computing on Database Query Optimization. In *Proceedings of the 2022 International Conference on Management of Data*. ACM, Philadelphia PA USA, 2512–2514. https://doi.org/10.1145/3514221.3520257

[36] Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. 2023. Quantum-Inspired Digital Annealing for Join Ordering. In *Proc. VLDB Endow.*, Vol. 16.

[37] Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. 2023. Quantum Optimisation of General Join Trees. In *VLDBW'23*.

[38] Ruslan Shaydulin, Hayato Ushijima-Mwesigwa, Ilya Safro, Susan Mniszewski, and Yuri Alexeev. [n.d.]. Network Community Detection on Small Quantum Computers. 2, 9 ([n. d.]), 1900029. https://doi.org/10.1002/qute.201900029

[39] Alfonso Shimbel. [n.d.]. Structural parameters of communication networks. 15, 4 ([n. d.]), 501–507. https://doi.org/10.1007/BF02476438

[40] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26, 5 (Oct. 1997), 1484–1509. https://doi.org/10.1137/S0097539795293172

[41] Mario Szegedy. 2004. Quantum Speed-Up of Markov Chain Based Algorithms. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS '04)*. IEEE Computer Society, USA, 32–41. https://doi.org/10.1109/FOCS.2004.53

[42] Dario Tamascelli and Luca Zanetti. [n.d.]. A quantum-walk-inspired adiabatic algorithm for solving graph isomorphism problems. 47, 32 ([n. d.]), 325302. https://doi.org/10.1088/1751-8113/47/32/325302

[43] V. A. Traag, L. Waltman, and N. J. van Eck. [n.d.]. From Louvain to Leiden: guaranteeing well-connected communities. 9, 1 ([n. d.]), 5233. https://doi.org/10.1038/s41598-019-41695-z

[44] Immanuel Trummer and Christoph Koch. 2016. Multiple query optimization on the D-Wave 2X adiabatic quantum computer. *Proceedings of the VLDB Endowment* 9, 9 (May 2016), 648–659.

[45] Immanuel Trummer and Davide Venturelli. 2024. Leveraging Quantum Computing for Database Index Selection. In *Proceedings of the 1st Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications (Q-Data '24)*. Association for Computing Machinery, New York, NY, USA, 14–26. https://doi.org/10.1145/3665225.3665445

[46] Valter Uotila. 2022. Synergy between Quantum Computers and Databases. *Proceedings of the VLDB 2022 PhD Workshop co-located with the 48th International Conference on Very Large Databases (VLDB 2022)* 3186 (Sept. 2022), 4.

[47] Valter Uotila. 2024. SQL2Circuits: Estimating Metrics for SQL Queries with A Quantum Natural Language Processing Method. arXiv:2306.08529 (June 2024). https://doi.org/10.48550/arXiv.2306.08529 arXiv:2306.08529 [quant-ph].

[48] Valter Uotila. 2024. Tensor Decompositions and Adiabatic Quantum Computing for Discovering Practical Matrix Multiplication Algorithms. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*. arXiv:2406.13412 [quant-ph] https://arxiv.org/abs/2406.13412

[49] Valter Uotila and Jiaheng Lu. 2023. Quantum Annealing Method for Dynamic Virtual Machine and Task Allocation in Cloud Infrastructures from Sustainability Perspective. In *ICDEW*.

[50] Lauri Vuorenkoski. 2024. Implementing graph algorithms on quantum annealers. (2024). https://helda.helsinki.fi/items/4b25a318-cdb0-44c2-b5a6-207c3782942e/URN:NBN:fi:hulib-202403131494

[51] Tobias Winker, Sven Groppe, Valter Uotila, Zhengtong Yan, Jiaheng Lu, Maja Franz, and Wolfgang Mauerer. 2023. Quantum Machine Learning: Foundation, New Techniques, and Opportunities for Database Research. In *Companion of the 2023 International Conference on Management of Data* (Seattle, WA, USA) *(SIGMOD '23)*. Association for Computing Machinery, New York, NY, USA, 45–52. https://doi.org/10.1145/3555041.3589404

[52] Tobias Winker, Umut Çalıkyılmaz, Le Gruenwald, and Sven Groppe. 2023. Quantum Machine Learning for Join Order Optimization using Variational Quantum Circuits. In *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments (BiDEDE), Seattle, WA, USA*. https://doi.org/10.1145/3579142.3594299

[53] Umut Çalıkyılmaz, Sven Groppe, Jinghua Groppe, Tobias Winker, Stefan Prestel, Farida Shagieva, Daanish Arya, Florian Preis, and Le Gruenwald. 2023. Opportunities for Quantum Acceleration of Databases: Optimization of Queries and Transaction Schedules. *Proc. VLDB Endow.* 16, 9 (2023), 2344–2353. https://doi.org/10.14778/3598581.3598603