Final Report

José Jesus

November 7, 2023

Abstract

In this project we addressed the problem of rail optimization, which has been showed to be equivalent to the problem of finding the maximum independent set of (MIS) a graph. Finding the MIS is a NP-Hard problem for classical computers but can be solved very efficiently using the Neutral atoms quantum computers of Pasqal. We translated this approach into a working algorithm and applied this algorithm to 2 examples to create working prototypes - namely a small grid and a train station. We also demonstrate energy advantages of finding the MIS of with a maximum degree bigger than 3 using quantum computers with more than 140 qubits. Finally, possible future Industrial applications were considered.

1 Introduction

Climate change is one of the defining challenges of the 21st century. Indeed, as the UN secretary general, Antonio Guterres, puts it, we are now in the age of Global Boiling [1]. Therefore, a Green transition for the global economy is desperately needed.

A key element of this Green Transition is Rail. The transportation sector accounts for around 20 % of global gas emissions [2], second only to the energy production industry. Therefore, the usage of low emission transports, such as Trains, are more needed than ever. However, these systems are not always utilized to their full potential. Indeed, who amongst us has never experienced the displeasure of delays or being stuck on a stopped train because some line is occupied?

Therefore, efficient usage of rail lines and systems is crucial for a healthy public transport sector and therefore to the green transition. However, the problem of scheduling is NP-complete [3] which makes it very hard to solve on classical computers. New strategies, such as quantum computing, are necessary to optimize the usage of an expanding rail infrastructure. This is the issue that we plan to tackle and where the new Neutral atoms Computers from Pasqal can provide a decisive advantage.

This way, the objective of this project is clear: To achieve a better and more efficient usage of current rail infrastructure through the usage of Quantum Computers to solve scheduling problems.

2 Problem definition

The idea of using quantum computers to optimize rail schedules is not new, as demonstrated by the works of [4, 5, 6], however so far the works presented are based on quantum annealing. However, as demonstrated in [7] and [8], the problem of rail scheduling can be reduced to the problem of finding a Maximum Independent Set [9] of a graph in which each vertices represents available slots for train movement and each edge represents conflicts between these slots.

This representation is very naturally encoded and solved in Pasqal neutral atoms machines [10]. However, no tests or experiments involving these quantum computers to optimize rail scheduling has been performed, as far as the author knows. Therefore, this problem is a great candidate for The Blaise Pasqal [Re]generative Quantum Challenge given its potential for the future green transition and solid foundation on existing research and was therefore chosen.

This way, we roughly follow the approach of [8] to reduce our scheduling problem to a Maximum Independent Set problem, which is then solved on a Pasqal Quantum Machine. In this framework, we have a set of slots, for each train. Each slot represents a path at a specific time that can only be occupied by 1 train and are pre-known (as train infrastructure is fixed). These are represented by vertices in our graph, and therefore will correspond to qubits in our implementation. When 2 slots are in conflict, by considering the same train or the same track at the same time, they are connected by an edge. This way, we can use the Rydberg Effect and an analogue algorithm [11, 12, 13] to slowly evolve our state into the ground state that will maximize the number of slots used, and therefore optimize the grid.

We focused specifically on freight trains because the slot representation of train paths is particularly well suited for these, given that to schedule a train a transport company has to make a train path application with an infrastructure manager, for instance "Deutsche Bahn" [14], which contains the departure and arrival points and will represent our slot, or could potentially be divided into smaller sub-slots depending on the path algorithm used by the infrastructure manager. Additionally, increasing freight transport of products is generally considered a key factor in reducing climate change, given that it produces less emissions than alternative shipping methods, i.e planes and trucks.

3 Algorithm

The following represents the description of the analogue algorithm:

- 1 - Given N trains with X possible slots in the grid for them, where each slot represents a valid path at a valid time, create the 2D graph where each vertice corresponds to a slot, X_i;
- 2 - For each conflicting slot, add an edge to the graph;
- 3 - Implement the graph on a Pasqal machine using *Pulser* - the coordinates for the vertices are decided by the required distance in order to establish edges through Rydberg Blockade;
- 4 - Using Global Rydberg pulse, start the system at the mixed state and slowly evolve to the ground state;
- 5 - Measure in the Rydberg basis;

In this first toy model, we are considering only freight trains that all work at the same speed and are only interested in going from the starting destination to the final, without stops. We are also considering that the entire track is occupied by a single train. These assumptions simplify our first implementation, but importantly do not change the quantum part of the algorithm - only the slots that are predetermined. These slots can be constructed in many different ways: for instance by taking the shortest paths algorithms, by taking the approach of [15] or any other path finding algorithm. The only quantum input is the resolution of conflicts arising from the generation of all the different slots.

Therefore this first prototype should give a clear view of the feasibility of the solution. To improve on it, to better describe real life conditions, we simply need to improve our slot formation, for instance by subdividing each journey into multiple slots.

4 Prototype

4.1 Simple Grid Example

To implement the algorithm, we first start with the Ising Hamiltonian H_Q:

$$H_Q = \sum_i^N \frac{\hbar\Omega(t)}{2}\sigma_i^x - \sum_i^N \frac{\hbar\delta}{2}\sigma_i^z(t) - \sum_{j<i} \frac{C_6}{|r_i - r_j|^6} n_i n_j \quad (1)$$

where by controlling the amplitude $\Omega(t)$ and the detuning $\delta(t)$ we can first initialize our register in a mixed state and slowly evolve it to the ground state of the system. The Rydberg term will ensure that no 2 connected edges will make part of the selected independent set and by minimizing the energy, i.e maximizing the Rydberg term, we can obtain the maximum independent set.

In this simple example we consider 3 freight trains, A, B and C. There are 3 possible slots for A's trip, 2 for B's and only 1 for C. C's slot needs to use the same track as one of A's and one of B's, and a possible slot of B also shares the same track with one of A's. The following figure gives an example of such a grid:

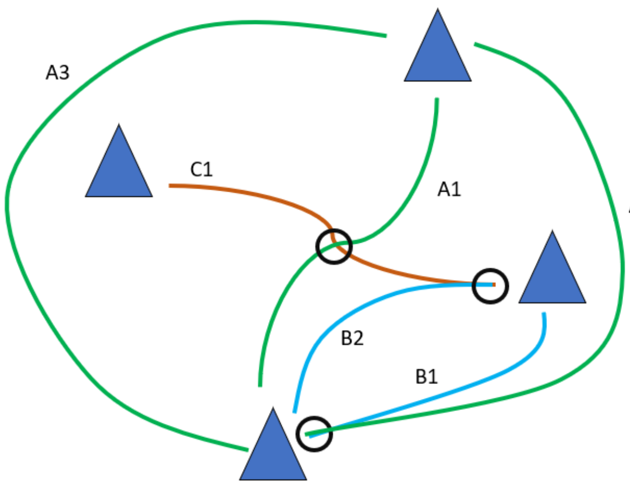

Figure 1: Small example grid where the triangles represent cities and the lines the available slots/paths. The green lines represent train A's slots, the blue lines B's and the brown line C. Conflicts are highlighted using circles

This can be translated into the following graph in *Pulser* using a Rydberg Blockade of 1 :

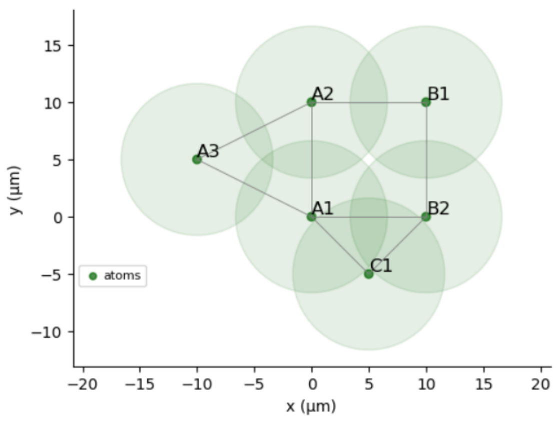

Figure 2: Graph corresponding to the simple grid

From here we can evolve adiabatically until we reach the ground state, just like shown in the webinars. We used the following pulse:

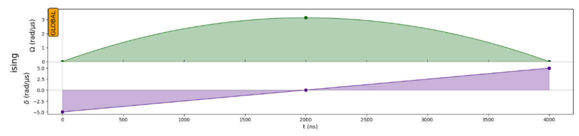

Figure 3: Pulse used for adiabatic evolution

The obtained result was:

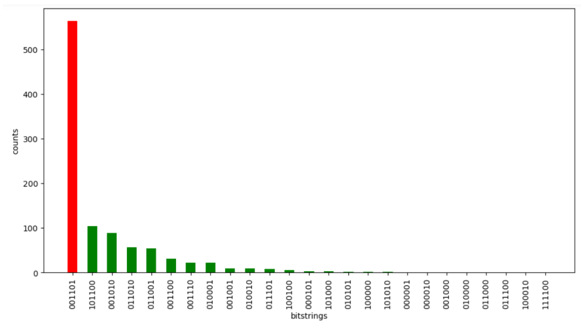

Figure 4: Measurement of final state

where we can see that the quantum computer correctly identifies the optimal schedule: by using slot $A3$, slot $B1$ and slot $C1$ all trains can run (which given the coding used corresponds to the bitstring 001101).

4.2 Train Station Example

Another possible and interesting application of the same algorithm consist of looking at only 1 station instead of a whole grid, which is also an important NP-problem [7], and is considered a key step in a more efficient rail usage. The problem is fundamentally the same, with the exception that now the slots represent times at which possible platforms will be occupied by incoming trains. Conflicts will arise when 2 or more trains need to occupy the same platform at the same time. The amount of time that a train occupies a certain platform is given by safety protocols and is also known. Given that the number of platforms and switches between platforms is limited, when compared to the rail infrastructure of a country, this problem shows more promise for early NISQ implementation.

We considered a toy train station model consisting of 5 possible routes for 4 trains. The trains are constrained due to their entry and exit point:

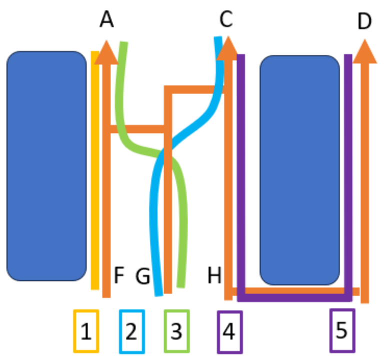

Figure 5: Small example Station where the colours represent paths that the trains can take.

In this example, the letters represent entry/exit points and the numbers represent possible paths. Each colour is associated to a specific train. We can see that only the purple train has 2 different possible paths and there is a conflict between the yellow train and the green, between the green and the blue and between the blue and 1 of the possible options of the purple. We can translate this onto the following graph:

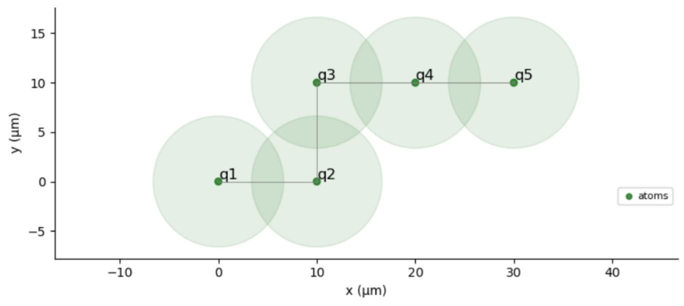

Figure 6: Representation of our toy station into a graph.

where qubit $q1$ represents the slot of the yellow train, $q2$ the slot for the blue train, $q3$ the slot for the green train, $q4$ a slot for the purple train and $q5$ the second slot for the purple train. Using the same global pulse to adiabatically evolve the system we get the final result:

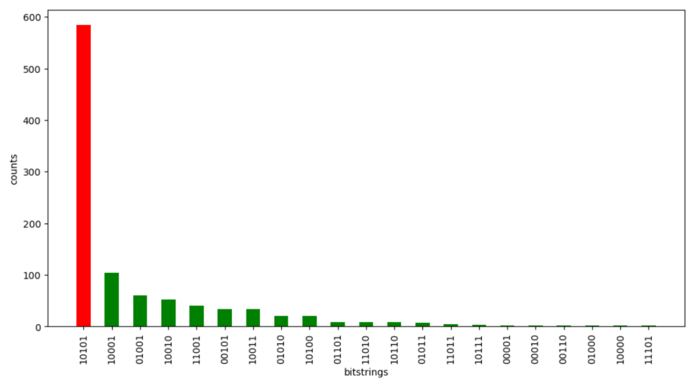

Figure 7: Final measurement for the Train Station Model Toy

The result is exactly as we expected, that is we select qubits 1,3 and 5 so the yellow, green and purple Train can enter the station but the green cannot. If we take into consideration a second time slot for the green train, with an extra qubit $q6$, we get the following graph:

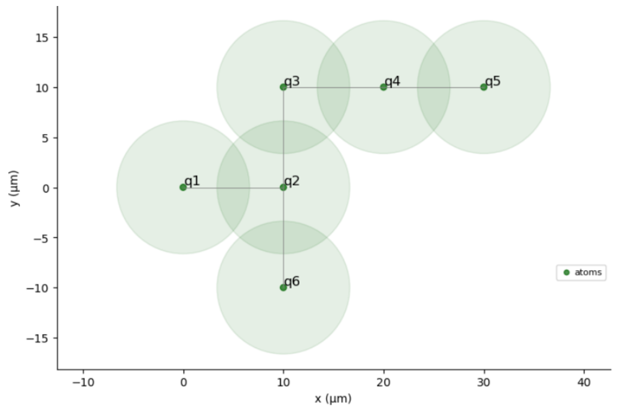

Figure 8: Representation of our toy station into a graph with an extra time slot for the green train.

we obtain the following final measurement:

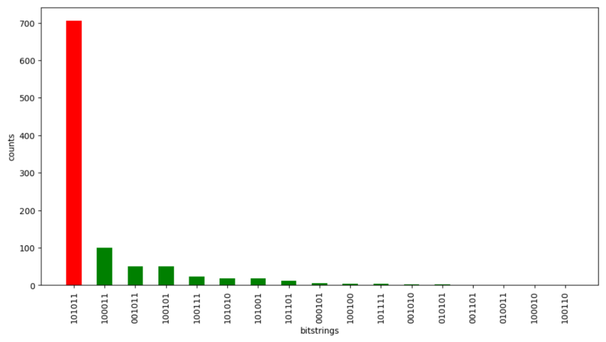

Figure 9: Final measurement for the Train Station Model Toy with an extra time slot for the green train

where we correctly identify via qubit q6 that the green train has to pass after all others.

With these 2 examples we demonstrate the capacity of our algorithm to tackle small scale problems which can then be scaled to practical situations.

5 Feasibility

5.1 Energy Analysis

As required by the Hackathon rules we studied the Energy requirements of our algorithm and compared it with classical algorithms that solve the same problem. To present a fair and balanced comparison we compare only classical algorithms that solve the problem of finding the Maximum Independent Set and disregard the problem of producing the slots (as that also has to be taken into consideration for our quantum approach). We also considered only algorithms capable of producing the exact solution to this problem,as we wanted to compare ”apples to apples”.

To that end the first task is to identify the cost in terms of Oracle calls (or computer operations) to solve the MIS problem using a classical computer. According to the works of Ref. [16], for a graph with n vertices a maximum degree of 3 (which means that each vertice has at maximum 3 edges) we need at least 1.0836^n oracle calls, for a graph with a maximum degree of 6 we need at least 1.1893^n oracle calls, for a graph with a maximum degree of 7 we need at least 1.1970^n oracle calls and for a graph with a degree above 7 we need a minimum of 1.1996^n oracle calls.

With this information we can now compute the expected computing time in hours and emissions for each of the graph types referenced above. We considered that the basic GPU server can produce an oracle call in 1 ms, which is in the same order of magnitude as the results in [17] and that Juliet-Curie server was 1000 times faster than that. For the 140 Qubits machine the maximum size of the graphs will be $n = 140$ (each qubit a vertice) and we considered that 3600 shots would be required to achieve a good result (using a worst case scenario 1 shot/s that gives an hour of quantum computing, which should ensure a rate of success above noise levels). Using these considerations we obtain:

Type of Graph	Quantum (h)	GPU Server (h)	Juliet Curie (h)
max degree 3	1	0.02	0.00002
max degree 6	1	9649.3	9.7
max degree 7	1	23816	23.8
max degree ¿ 7	1	32270	32.3

which has a cost in tonnes of CO_2 of :

Type of Graph	Quantum	GPU Server	Juliet-Curie
max degree 3	0.00361	2×10^{-6}	4.6×10^{-6}
max degree 6	0.00361	0.8	2.1
max degree 7	0.00361	2.1	5.1
max degree ¿ 7	0.00361	2.8	6.9

with this example we can clearly see the advantage of quantum computers for hard classical problems. Given that the algorithm is the same regardless of the number of edges in the graph we can expect the same computational time (although there should be a slight increase in energy costs for reproducing the geometry of more complex graphs). However that is not the case for computational computers and we can see that there is a clear energy advantage with 140 qubit system for the more complex types of graphs considered.

If we instead look into the system with a 1000 qubits this advantage would even be more dramatic. For a graph with a max degree of 3, with $n = 1000$ we would need 7×10^{34} oracle calls. This gives an absurd computing time. Even if we considered that the GPU server could implement an Oracle each picoseconds, which is completely unfeasible, we would still have a computing time of more than 2×10^{19} hours which would be an ecological disaster. Considering that for the quantum machine it would be exactly the same adiabatic algorithm, it is reasonable to consider that the required time would still be in the tenths of hours.

It is then clear for these simple examples the need for quantum computing, not only as a new computational resource to solve hard problems, but also as a green computational resource.

5.2 Industrial Applications

So far we considered only the exact solution of the MIS problem created from a graph representing the available slots for trains to use. However, that does not mean that an incomplete, but good solution is not known. Indeed, by using heuristics the authors of [8] manage to extract solutions to problems that would require more than 80000 qubits (graphs with 80000 vertices), pertaining Germany's train network, which is beyond the NISQ era in only 6 hours of computing time. Therefore, we can see that there is still a long way to go for useful, practical advantages from using Quantum Computers for rail scheduling at the country level. The obvious disadvantage is that by using this approach we can not be certain that we are using our the available resource to their utmost potential. Indeed, the authors of Ref. [8] found out in their paper that they could schedule an extra 100 trains when compared to the next best available commercial algorithms. This suggests that there is still room for improvement - especially considering that this is a lower bounded value on the size of the MIS.

However, at the station level, the graphs only need around 3000 qubits [7]. This number is feasible in the next few years and could potentially be a practical application of quantum computing. Additionally, when bench-marking the algorithm in [8], for graphs with vertices between 1024 and 4096 and low density (ratio of edges present and maximum number of possible edges) - see table 2 of the corresponding paper - the authors could only give upper and lower bounds on the size of the maximum independent set and not an exact answer. This further strengthens the claim that the next quantum computers with a few thousand qubits will be able to solve interesting problems that current classical computers cannot. This is especially true given that new capacities, like 3D graphs are being worked on which will allow even more geometries (which could be a potential roadblock) to be embedded in Pasqal machines.

6 Conclusion

This first prototype demonstrates the potential that Quantum Computers to solve hard computational problems and power the green transition. However, these are still toy models and there is still a long distance to practical industrial applications.

Nonetheless, these still shine light on key advantages that quantum computing brings for the green revolution. Indeed, beyond the advantage that we are trying to bring with this project of optimizing the usage of rail, which is a low carbon form of transportation, we also demonstrated that the calculation itself is greener using quantum computation instead of classical computation.

It is also important to notice that although we focus on rail scheduling, solutions to scheduling problems and conflict management problems have a plethora of different applications, like runways optimization in airports and machine allocation optimization in factories, which further strengths the argument that quantum computers truly are a revolutionary technology.

Finally, the team would like to thank Pasqal for organizing this Hackathon which allowed us to learn more about a different type of computing and addressing interesting, real world problems. All the code used for this project can be found in: [18]

References

[1] Antonio guterres decleration's. https://www.cnbc.com/2023/07/27/the-era-of-global-boiling-has-arrived-says-un-boss-antonio-guterres.html. Accessed: 06-11-2023.

[2] https://www.statista.com/topics/7476/transportation-emissions-worldwide/. Accessed: 06-11-2023.

[3] Jeffrey D. Ullman. Np-complete scheduling problems. *Journal of Computer and System sciences*, 10(3):384–393, 1975.

[4] Krzysztof Domino, Akash Kundu, Özlem Salehi, and Krzysztof Krawiec. Quadratic and higher-order unconstrained binary optimization of railway rescheduling for quantum computing. *Quantum Information Processing*, 21(9):337, 2022.

[5] Mátyás Koniorczyk, Krzysztof Krawiec, Ludmila Botelho, Nikola Bešinović, and Krzysztof Domino. Application of a hybrid algorithm based on quantum annealing to solve a metropolitan scale railway dispatching problem. *arXiv preprint arXiv:2309.06763*, 2023.

[6] Krzysztof Domino, Mátyás Koniorczyk, Krzysztof Krawiec, Konrad Jałowiecki, Sebastian Deffner, and Bartłomiej Gardas. Quantum annealing in the nisq era: railway conflict management. *Entropy*, 25(2):191, 2023.

[7] Peter J Zwaneveld, Leo G Kroon, H Edwin Romeijn, Marc Salomon, Stephane Dauzere-Peres, Stan PM Van Hoesel, and Harrie W Ambergen. Routing trains through railway stations: Model formulation and algorithms. *Transportation science*, 30(3):181–194, 1996.

[8] Julian Reisch, Peter Großmann, Daniel Pöhle, and Natalia Kliewer. Conflict resolving–a local search algorithm for solving large scale conflict graphs in freight railway timetabling. *European Journal of Operational Research*, 293(3):1143–1154, 2021.

[9] Robert Endre Tarjan and Anthony E Trojanowski. Finding a maximum independent set. *SIAM Journal on Computing*, 6(3):537–546, 1977.

[10] Sepehr Ebadi, Alexander Keesling, Madelyn Cain, Tout T Wang, Harry Levine, Dolev Bluvstein, Giulia Semeghini, Ahmed Omran, J-G Liu, Rhine Samajdar, et al. Quantum optimization of maximum independent set using rydberg atom arrays. *Science*, 376(6598):1209–1215, 2022.

[11] Igor I Ryabtsev, Denis B Tretyakov, and Ilya I Beterov. Applicability of rydberg atoms to quantum computers. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 38(2):S421, 2005.

[12] M Morgado and S Whitlock. Quantum simulation and computing with rydberg-interacting qubits. *AVS Quantum Science*, 3(2), 2021.

[13] Sam R Cohen and Jeff D Thompson. Quantum computing with circular rydberg atoms. *PRX Quantum*, 2(3):030322, 2021.

[14] https://fahrweg.dbnetze.com/fahrweg-en/customers/services/Train-paths/train_path_application. Accessed: 06-11-2023.

[15] Florian Dahms, Anna-Lena Frank, Sebastian Kühn, and Daniel Pöhle. Transforming automatic scheduling in a working application for a railway infrastructure manager. In *Rail Norrköping Conference*, 2019.

[16] Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. *Information and Computation*, 255:126–146, 2017.

[17] https://cg.ivd.kit.edu/downloads/GPUComputing_assignment_2.pdf. Accessed: 06-11-2023.

[18] https://github.com/JoseDCJesus/PasqalHackathon/tree/main. Accessed: 07-11-2023.