POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Quantum paths finding algorithm

Relatore
prof. Bartolomeo Montrucchio

Davide Integlia

April 2022

Summary

The aim of this work is to design a hybrid quantum-classical algorithm to solve an optimization problem in the traffic flow domain. The basic idea is to spread vehicles on different paths in order to make the traffic as fluent as possible, while also minimizing the exposure to air pollution along the path. The problem is represented as Quadratic Unconstrained Binary Optimization (QUBO), that is a combinatorial optimization problem, frequently used for computer science applications. Such QUBO formulation can be mapped on a Quantum Annealer, a special machine that performs Adiabatic quantum computation. The problem is thus formulated: Given two distinct points in a city, a defined number of paths are selected among a set generated by Open Street Maps APIs. These paths are selected using a classical greedy algorithm, that results in a compromise between computational time and quality of the solution. At this point a basic QUBO matrix is built starting from the description in the foundational paper for this domain (Volkswagen and D-Wave, 2017). Generated matrixes are perturbed in two parts:

- The former part takes in consideration the coexistence time of two or more cars on the same street segment. For each car it is also considered the departure time. To simplify the problem a constant velocity is considered for all cars.
- The latter part performs a pollution evaluation as malus of the optimization. One is applied to each path to take in consideration the cost of pollution level all along the way and another for each path couples that consider the pollution cost only for common street segments.

The formulated problem is solved on a D-wave quantum annealer and QPU parameters are properly tuned to ensure a good and appreciable solution.

Acknowledgements

I thank my family and everyone who supported me and made me the person I am today.

Contents

Chapter 1

Introduction

1.1 Global pollution

From the first hints of interest on the part of man to the ambitions of the integral ecology and related issues arising from environmental pollution is revealed that nature cannot be considered simply as a mere frame of human life. In fact, the first thing that comes to mind when pronouncing the term "pollute" are the damages that this phenomenon determines and, between these, those perceivable by the human being.

Consequently, environmental pollution is intended as a total upheaval, often irreversible for entire territories, but also of the entire planet. From the point of view of the causes that can determine this phenomenon, we know the one resulting from the introduction of harmful substances into the environment due to natural events. The eruption of a volcano for example pollutes the environment and damages human health and in some cases releases into the environment such quantities of substances that exceed the digestion capacity and absorption of the environment itself. But even if significant, these circumstances constitute a negligible portion of the concomitant causes that are determining the worsening of the problem. Most of the polluting elements introduced in the environment derive from the carrying out of activities, lawful, tolerated and / or illegal, man-made. Indeed, natural phenomena and human behavior can also determine the modification of elements already present in nature up to constituting a deterioration of environmental conditions. Those implications are what we know as thermal, acoustic, electromagnetic, atmospheric, water pollution, marine, etc.

Among the actors that have a significant responsibility in determining the increase of pollutants level, there are the industries. Indeed as reported[18], during the production cycle they release in the environment "substances that today endanger life and health of over 95 million people and who are present in populated areas in significantly higher quantities than in the past. The characteristics of these substances have been extensively studied and documented, clearly proving their toxicity". The quoted Environmental organization identifies these substances in the following: hexavalent chromium; lead; mercury; pesticides; radionuclides; cadmium.

It follows that industrial activities combined with those carried out during daily activities cause gas emissions in the atmosphere. This has now become a problem on a global scale, causing in little more than a century the current characterized conditions of the greenhouse effect, smog and acid rain.

Also, the great settlements and human activities of our day have an enormous impact on the environmental balance of the entire planet, creating the conditions that could lead to a deep climate change. The dynamics that are the contributing causes of the aforementioned types of pollution, can be countered, on the one hand, by adopting different production models, based on reuse, recycling, limited use of non-renewable resources and on the other, by significantly modifying the human behaviors such as to determine at least the containment of environmental pollution process.

Concerns represented by the scientific community related to climate change induced by pollution, have led the United Nation to adopt rules to counteract the phenomenon and prevent damages to the planet resulting from the increase of temperatures. As part of the first COP (Conference of the parties)[19], at Kyoto Protocol[20] and at the subsequent conferences on the subject, it was identified what is considered one of the major causes of air pollution: The movement of petrol cars. Indeed, combustion emits different compounds from the exhaust, in particular the fine powders Pm10 and Pm 2.5, extremely harmful for the environment. Consequently, the modification of the air brought by emissions of gas, fine dust and fumes, is extremely harmful to human health. It follows that by reducing the circulation of cars with thermal engines and / or optimizing the circulation of the aforementioned vehicles, we can at least contain the phenomenon of environmental pollution that derives from it. Faced with the need to respect the commitments undertaken by the States participating in the various climate assemblies and the constant need to move of the man, while trying in any case to reduce the introduction of pollutants into the environment, industry efforts should be rewarded for the production of electric cars. In this regard, it is useful to point out that the birth of the first electric car took place in the same period of time as the petrol cars. Although the battery electric car (BEV, also called "electromobile") at the beginning of the 20th century, was one of the first types of car to be invented, tested and marketed. The improvement of the batteries, thanks to the work of the French Gaston Plante in 1865 and Camille Faure in 1881, allowed the flourishing of electric vehicles[21]. In the late 1800s, before the preponderance of the powerful but polluting internal combustion engine, electric cars held many records for speed and distance traveled on one charge. In fact, electric cars in addition to having the advantage of not introducing pollutants into the environment, have the further positive aspect, not to be underestimated, represented by the silence of their engine. However[21], the performance gap soon will be filled by petrol cars (the main cause is attributable to the weight and volume of the batteries, which are difficult to transport during journeys). The issue of autonomy reflects a problem that is still current, despite the fact that there have been numerous improvements (although now electric models allow you to travel more kilometers than old electric prototypes); in addition, it is necessary to consider the long recharge times, the limited autonomy of the batteries and the deterioration, even if with the advancement of research new types of rechargeable batteries and new technologies have increased their autonomy and life, reducing the charging time at the same time. Furthermore, in most cases electric vehicles do not lead to a real zeroing of CO2 levels due to the fact that most of the countries producing electricity obtain it thanks to thermal power plants that burn fossil fuels. Despite these disadvantages, they can offer numerous benefits especially from an environmental point of view thanks to the absence of emissions of pollutants. The functioning of electric cars requires that they use the chemical energy stored in the energy tank consisting of one or more rechargeable batteries; the low consumption, the very high performances, the silence and the pollution close to zero, make the electric car, according to the prevailing thought, a product capable of saving the car market and in a way to respect the international agreements of Paris climate conference held in 2015.

From 1900[1], CO2 emission continue to increase at each year, except for the ones in which human history was signed, such: World war, financial crisis, COVID pandemic and more. Following is reported a graph that show the incremental CO2 levels over years, taking as unit of measurement GtCO2, for instance gigaton of carbon dioxide.

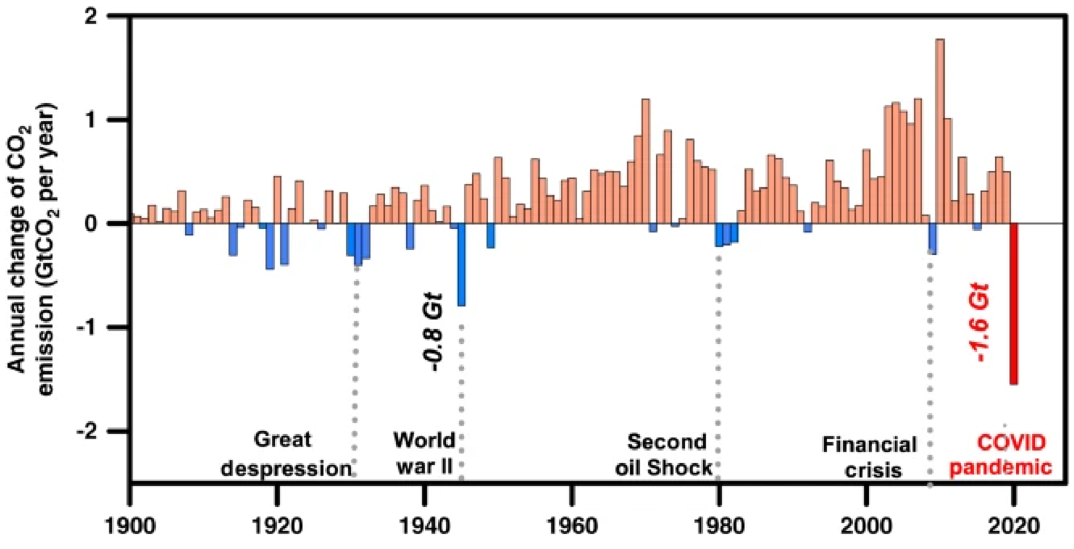


Figure 1.1. CO2 released in the environment over years

1.2 Quantum computers

Quantum computers[17] have been theorized in '80 decades of the previous century and they represent the key to overcome the actual computational power provided by super computer. Quantum computers exploit the quantum mechanics law to perform operation in a negligible time, compared to current methodology. Classical computers save information in RAM in the form of presence or absence of electrical power, instead quantum computers use qubits namely quantum bits. Differently from classical computers, the quantum ones do not use a linear logic based on sequences of zero and one, in fact its velocity depends on the new methodology used to analyze and process data. This new approach is not applicable to every single situation, however quantum computers have a lot of application fields, it is able to evaluate in few seconds a solution that would require thousands of years. Increasing interest about quantum computers raise thanks or due to the boundary of Moore's law ,since the power of a microprocessor is proportional to the number of microtransistors of which is composed, it states that this number double every 18 months, but this law started to lose validity due to mechanics limits. For this reason[17] scientists tried to find alternative ways to overcome the problem of computational power, and the answer was found in quantum computers. Although they are in an embryonic state, due to the absence of a standard and the scarcity of people able to work on this field, but they are nowadays usable and a lot of projects born around this new technology included this work. Qubits represent[17] a huge turning thanks to their quantum property, since they can assume the equivalent zero and one states simultaneously. This allows to manage the same amount of information using less memory, this allows to work in parallel and find multiple solutions from just one problem without following a linear logic sequence.

1.2.1 Functioning of a quantum computer

The real obstacles[17] regarding the operation of quantum computers is the management of the qubits, even now it is possible to handle just few quantum bits. As mentioned before quantum logic is based mainly on two principles: Superpositions and Entanglement. Have been identified two methods to manage qubits:

- The first method plans to cooling the circuit, in order to create a superconductor in which electricity transits without resistance. We can call them "quantum points" that are nano-structures inside of a larger semiconductor with a larger energy interval.
- The second method is based on trapped iones, particles with charge, surrounded by an electromagnetic field, which perturbs the particles such that they can work as qubits.

Even if it is possible to already use quantum computers to solve some kind of problems, remains the need to found and build an adequate hardware structure and develop dedicated software to manage all the system.

1.2.2 Application fields and consequence

Using the computation power of quantum computers[17], we are going to perform more and more complex simulations in the biology field and all of its sub-branches. It will be possible to find a solution for a lot of disease in less time respect to actual schedule and this can save thousand of lives. Another important field is the one that concerned the cryptography, since it will be possible to break in and easiest way the crypted codes, leading to a crash of all cyber-system such banks, privacy, national defense, spying and more. It is obvious that "from great powers comes great responsibilities", so there is the need, in the near future, to regulate who and in which situation quantum computers can be used.

1.2.3 Quantum computers deployment situation

The first company[17] who first invested in quantum computers was IBM, but actually, the supremacy of quantum computers is held by a collaboration between NASA an Google with a more than 5000 qubits computers[3], assembled in an architecture called Pegasus, currently located at Quantum Artificial Intelligence Lab in California. This computer is built along the line of its predecessor that mount a Chimera processor and presents the following improvements, as shown by hpcwire[3]:

- New Topology: Pegasus is the most connected of any commercial quantum system in the world. Each qubit is connected to 15 other qubits (compared to Chimera's 6), giving it 2.5x more connectivity. It enables embedding of larger problems with fewer physical qubits. The D-Wave Ocean software development kit (SDK) includes tools for generating the Pegasus topology. Interested users can try embedding their problems on Pegasus[3].
- "Lower Noise: next generation system will include the lowest noise commercially-available quantum processing units (QPUs) ever produced by D-Wave. This new QPU fabrication technology improves system performance and solution precision to pave the way to greater speedups[3].
- "Increased Qubit Count: with more than 5000 qubits, the next generation platform will more than double the qubit count of the existing D-Wave 2000Q. Gives programmers access to a larger, denser, more powerful graph for building commercial quantum applications[3].
- "Expansion of Hybrid Software and Tools: Investments in ease-of-use, automation and provide a more powerful hybrid development environment building upon D-Wave Hybrid. Allows allowing developers to run across classical and the next-generation quantum platforms in Python and other common languages. Modular approach incorporates logic to simplify distribution, allowing developers to interrupt processing and synchronize across systems to draw maximum computing power out of each system[3].

Figure 1.2. D-wave Pegasus quantum computer

Chapter 2

Background

2.1 Quantum annealing

Quantum annealing[10] is a quantum technology that allows the solution of some kind of optimization problem using quantum computers. It is a practice that is expanding like wildfire thanks to studies made in the field of quantum physics. There are different approaches to perform quantum solving, but our focus is on quantum annealers, they represent a less flexible architecture but it is more stable and simple to build. Quantum annealing approach is based on four main physical concepts:

- Energy
- Superposition
- Quantum tunneling
- Entanglement

Quantum physics[10] is based on the De Broglie's hypothesis claiming that each particle has an associated wave function that is inversely proportional to its mass, this behavior is obvious thanks to the experiment of the double-slit. Nowadays, scientists are trying to figure out at which dimensions quantum physics laws cease to be valid, in this regard Markus Arndt and its team of Vienna's university have led an experiment on a molecule of 20000 atoms and they have verified that for seven milliseconds an interference happens. This is enough to state that the law of quantum mechanics are valid also for molecules of that dimension.

2.1.1 Energy

Since this paradigm is applied to optimization problems with a multitude of solutions, comes the need to define in which way a solution is better then another one. Here the energy concept is helpful, to each solution is associated an energy value that is much lower the better the solution is. It is immediate to pick and sort the solutions in function of ascending energy value, and consequently ascending objective function value.

2.1.2 Superposition

Superposition[11][13] is the first law of quantum mechanics, it states that given the dual nature wave-particles it is possible to sum two particles states, as performed for the wave in classic physics, and obtain a resultant valid state.

The wave function, in quantum physics, represent a state of the quantum system, it is a complex function dependent on spatial and temporal coordinates. It is the solution of Schrödinger's equation in which the square module represent a probability to find the particle in a specific location.

This principle[12] allows to interfere with the qubits using an electromagnetic field, this is the manner in which malus are applied. To get a visual match it is possible to draw an energy diagram that at first presents itself as a parable in which the lower position represent a lower energy level. Once annealing is applied, it splits the state forming a double-well potential, in which hollows are perturbed by electromagnetic field.

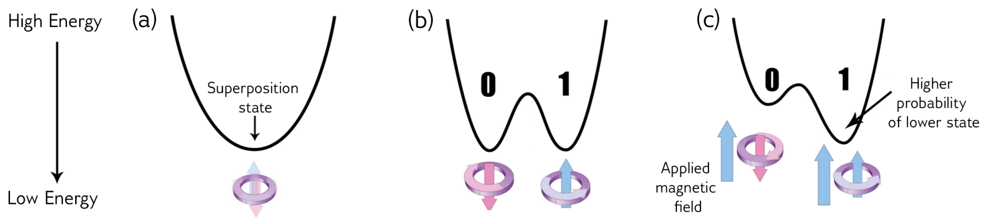


Figure 2.1. Energy diagram changes over time as the quantum annealing process runs and a bias is applied[12].

2.1.3 Entanglement

Quantum entanglement[10] is a concept relegated to quantum physics and not reducible to the classical one. This phenomenon claims that a state of a system doesn't depend only on the energy state of single particles, but on the superposition of all the states. Therefore particles are in some ways linked to each other since the behavior of the single, determines the behavior of the group. The incredible thing of this phenomenon is that particles don't need to be "touching", there is a sort of distance correlation without any spatial limits.

Using quantum annealing, malus are assigned by increasing the energy level parameter of qubits couples. This way, utilizing the concept of entanglement, each qubits perturbs others states based on the energy level that it owns and it is possible to create an energy map, that conceptually is equal to a gradient map. High levels of energy represent a repulsive gradient, lower levels of energy represent an attractive gradient.

2.1.4 Quantum tunneling

This is a feature[14] that allows qubits to switch to a state from another instantaneously, this means that particles must not spend other energy or time to overcome an obstacle. This is one of the main reason why QPUs are faster than traditional CPUs. A recent experiment explains in which way this phenomenon takes place: ”If the particle confronts an energy barrier, this encounter modifies the spread of the wave function, which starts to exponentially decay inside the barrier. Even so, some of it leaks through, and its amplitude does not go to zero on the barrier’s far side. Thus, there remains a finite probability, however small, of detecting the particle beyond the barrier. [...] Careful analysis revealed that it was, mathematically speaking, the peak of the tunneling photons’ wave functions (the most likely place to find the particles) that was traveling at superluminal speed. The leading edges of the wave functions of both the unimpeded photon and the tunneling photon reach their detectors at the same time, however—so there is no violation of Einstein’s theories of relativity.”. As reported, the tunneling is not immediate and is instead performed at the speed of light, but relative to electrical speed, the time spent to perform it is negligible.

2.2 QUBO problem

Combinatorial optimization[15] represents a key feature for a lot of applications, such as science, finance and more. Due to the nature of combinatorial optimization it can be long to extrapolate the global optimum, in the majority of cases it is take a good local optimum to avoid the high computational times. In those years[15] a mathematical formulation was found that is able to cover a large pool of combinatorial optimization problems, known as QUBO formulation. QUBO is an acronym an stands for ”Quadratic Unconstrained Binary Optimization”, its formulation is really simple and is able to efficiently solve complex existing combinatorial optimization problems. Here comes the necessity to transform or reformulate a problem in QUBO form, this can be achieved following some basic rules that will be treated after this chapter. QUBO formulations are taken into consideration thanks to their affinity to ising problem, that are the one used by QPUs to perform quantum annealing and optimization. Note that as far as the QUBO formulation is a powerful tool it does not provide a global optimum solution, but thanks to modern metaheuristic methods it is possible reach an high quality solution in a reasonable amount of time.

As pointed out[15] in Kochenberger and Glover (2006), the QUBO model encompasses the following important optimization problems:

- Quadratic Assignment Problems
- Capital Budgeting Problems
- Multiple Knapsack Problems
- Task Allocation Problems (distributed computer systems)
- Maximum Diversity Problems
- P-Median Problems

- Asymmetric Assignment Problems
- Symmetric Assignment Problems
- Side Constrained Assignment Problems
- Quadratic Knapsack Problems
- Constraint Satisfaction Problems (CSPs)
- Discrete Tomography Problems
- Set Partitioning Problems
- Set Packing Problems
- Warehouse Location Problems
- Maximum Clique Problems
- Maximum Independent Set Problems
- Maximum Cut Problems
- Graph Coloring Problems
- Number Partitioning Problems
- Linear Ordering Problems
- Clique Partitioning Problems
- SAT problems

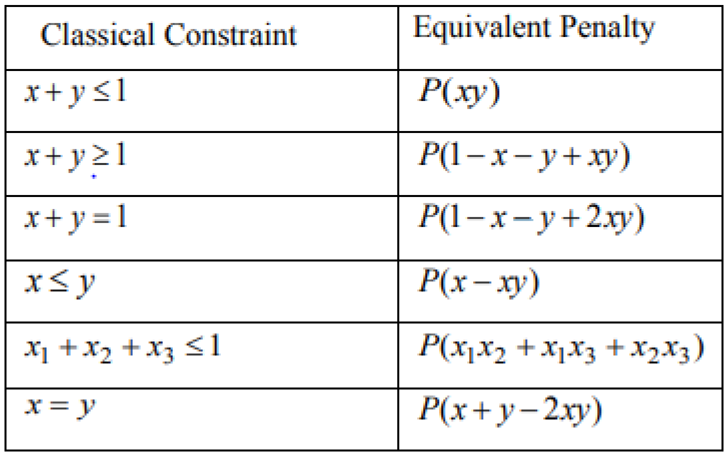
Now we're going to explain the way in which a QUBO problem can be formulated and in which constraints of different nature can be applied to it. QUBO model are formulated as follow:

$$minimize/maximize(x^t * Q * x)$$

Where x is a vector of binary decision variables and Q is a square symmetric or upper triangular matrix containing the objective function formulation and the constraints.

Starting from defined QUBO matrix[15], it can be perturbed to take in consideration the constraints of the problem. In particular, each constraints can be formulated as a quadratic penalties which values are added to QUBO cells to create an augmented objective function. Each penalty value P must be positive and must be chosen sufficiently large to assure the penalty term is indeed equivalent to the classical constraint, but in practice an acceptable value for P is usually easy to specify. To facilitate the choice of the P value is reported a useful table:

However, some problems require that P is a scalar penalty and for this kind of problems there is not a value that is perfect. Scalar penalty must be large enough to significantly asserts is presence, but not to much to not distort the pull of optimal



Classical Constraint	Equivalent Penalty
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y + xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \leq y$	$P(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1 x_2 + x_1 x_3 + x_2 x_3)$
$x = y$	$P(x + y - 2xy)$

Figure 2.2. Table of a few Known constraint/penalty pairs[15].

solution. It is necessary also consider the strength of the constraints, if it is soft or hard, because the second one must be tuned with attention to not overcome the objective function. In general, there is a ”Goldilocks region” from which it is possible to pick a good and feasible penalty value. Once an estimation of the final objective function value is performed, it is possible to define the penalties region as [75, 150]% of the estimated value.

2.2.1 Volkswagen QUBO formulation

Work treated in those pages is an extension of ”Traffic flow optimization using a quantum annealer”[16], so following is explained in which way they manage the problem as QUBO one, only and specifically for traffic flow. Their goal is to minimize the traffic congestion evaluating three possible paths for each car, if initial and final location are the same paths pool for each car can differ. The Q matrix of the problem as size [n*m, n*m] where n is the number of cars and m is the number of paths, in this case three for each car. It was assigned a binary variables for each pairs car-path and create a consistency constraint such that the sum of variables value belonging to a car was one and the sum of all variables are equal to the number of cars.

$$(\sum_{j=1}^{j=m} q_{i,j} - 1)^2 = 0$$

Further, it was defined the cost function that takes in consideration the common segment among paths:

$$cost(s) = (\sum_{i} \sum_{j} \sum_{s \in S_j} q_{i,j})^2$$

Where S_j is the set of common street segment.

At this point, put all together is obtained the objective function:

$$Obj = \sum_{s \in S} cost(s) + \lambda * \sum_{i} (\sum_{j=1}^{j=m} q_{i,j} - 1)^2)$$

Obtaining the so called LASSO problem. To retrieve the value of λ they find the maximum number of times some car i is present in cost functions cost(s), and use this value as λ.

Since the QUBO problem is formulated[16], it is necessary to fill the matrix Q, which values are retrieved applying this values when two path share the same street segment:

- We add a (+1) at diagonal index I(i, j) for every car i proposed with route j containing segment s.
- We add a (+2) for every pair of cars i1 and i2 taking route j containing segment s at the off-diagonal index given by indices I(i1, j) and I(i2, j).

And to constraint a car to choose only one routes:

- For every car i with possible route j, we add (-λ) to the diagonal of Q given by index I(i, j).
- For every cross-term arising from the first defined constraints, we add (2λ) to the corresponding off-diagonal term.

2.3 D-wave environment

Founded[8] in 1999, D-Wave is the leader in the development and delivery of quantum computing systems, software, and services and is the world's first commercial supplier of quantum computers. D-Wave's innovation and commitment to advancing the science of quantum computing has resulted in over 200 U.S. patents granted and over 100 peer-reviewed papers published in leading scientific journals.

The D-Wave[9] quantum processing unit (QPU) is a lattice of interconnected qubits. While some qubits connect to others via couplers, the D-Wave QPU is not fully connected. Instead, the qubits of D-Wave 2000Q and earlier generations of QPUs interconnect in a topology known as Chimera while Advantage QPUs incorporate the Pegasus topology. Some small number of qubits and couplers in a QPU may not meet the specifications to function as desired. These are therefore removed from the programmable fabric that users can access. The subset of the Pegasus or Chimera graph available to users is the working graph. The yield of the working graph is the percentage of working qubits that are present.

2.3.1 Chimera graph

Quantum problems[9] in this work are solved using this type of QPU structure since it is the only that supports the tuning of the parameter linked to the Boltmann constant.

In the D-Wave 2000Q and earlier systems, qubits are "oriented" on the QPU vertically or horizontally as shown in Figure 2.4.

For QPUs with the Chimera topology it is conceptually useful to categorize couplers as follows:

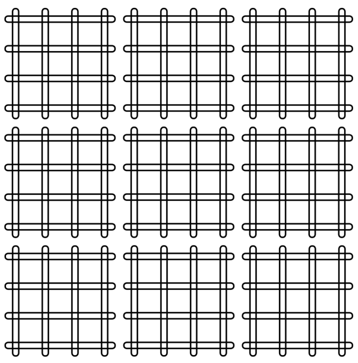


Figure 2.3. Qubits represented as horizontal and vertical loops. This graphic shows three rows of 12 vertical qubits and three columns of 12 horizontal qubits for a total of 72 qubits, 36 vertical and 36 horizontal[9].

Internal couplers

Internal couplers[9] connect pairs of orthogonal (with opposite orientation) qubits as shown in Figure 2.5. The Chimera topology has a recurring structure of four horizontal qubits coupled to four vertical qubits in a K4,4 bipartite graph, called a unit cell.

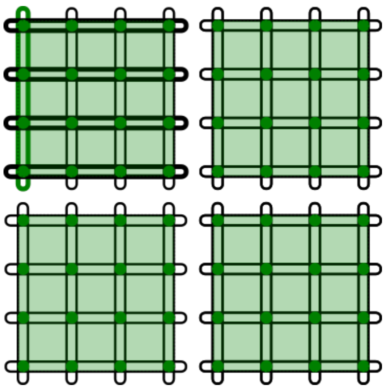


Figure 2.4. Green circles at the intersections of qubits signify internal couplers; for example, the upper leftmost vertical qubit, highlighted in green, internally couples to four horizontal qubits, shown bolded. The translucent green squares provide a helpful way to envision a recurring structure of this topology: a division of couplings into unit cells of K4,4 bipartite graphs[9].

A unit cell is typically rendered as either a cross or a column as shown in Figure 2.6.

External couplers

External couplers[9] connect colinear pairs of qubits—pairs of parallel qubits in the same row or column—as shown in Figure 2.7.

The K4,4 unit cells formed by internal couplers are connected by external couplers as a lattice: this is the Chimera topology. Figure 2.8 shows two unit cells that

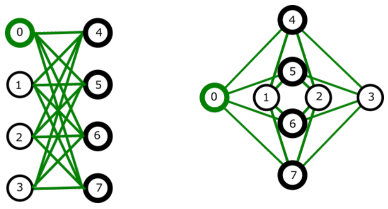


Figure 2.5. Chimera unit cell. In each of these renderings there are two sets of four qubits. Each qubit connects to all qubits in the other set but to none in its own, forming a K4,4 graph; for example, the green qubit labeled 0 connects to bolded qubits 4 to 7[9].

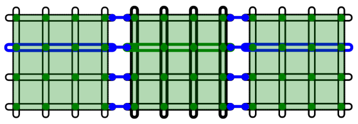


Figure 2.6. External couplers, shown as connected blue circles, couple vertical qubits to adjacent vertical qubits and horizontal qubits to adjacent horizontal qubits; for example, the green horizontal qubit in the center couples to the two blue horizontal qubits in adjacent unit cells. (It is also coupled to the bolded qubits in its own unit cell by internal couplers[9].)

form part of a larger Chimera graph.

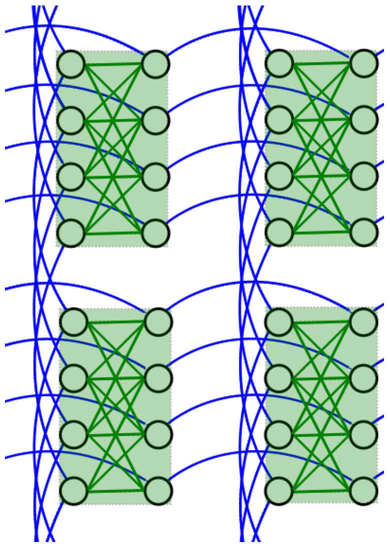


Figure 2.7. A cropped view of two unit cells of a Chimera graph. Qubits are arranged in 4 unit cells (translucent green squares) interconnected by external couplers (blue lines)[9].

Chimera qubits are characterized as having:

- nominal length 4—each qubit is connected to 4 orthogonal qubits through internal couplers.
- degree 6—each qubit is coupled to 6 different qubits

The notation CN refers to a Chimera graph consisting of an NxN grid of unit cells. The D-Wave 2000Q QPU supports a C16 Chimera graph: its more than 2000 qubits are logically mapped into a 16x16 matrix of unit cells of 8 qubits.

2.3.2 Pegasus graph

In the Pegasus topology[9], qubits are "oriented" vertically or horizontally, as in Chimera, but similarly aligned qubits are also shifted, as illustrated in Figure 2.9.

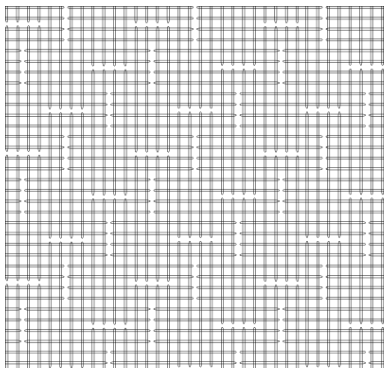


Figure 2.8. A cropped view of the Pegasus topology with qubits represented as horizontal and vertical loops. This graphic shows approximately three rows of 12 vertical qubits and three columns of 12 horizontal qubits for a total of 72 qubits, 36 vertical and 36 horizontal[9].

For QPUs with the Pegasus topology it is conceptually useful to categorize couplers as internal, external, and odd. Figure 2.10 and Figure 2.11 show two views of the coupling of qubits in this topology.

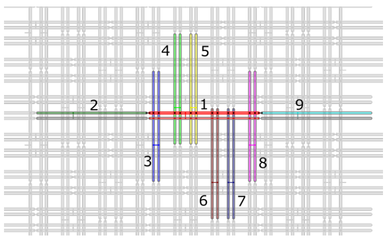


Figure 2.9. Coupled qubits (represented as horizontal and vertical loops): the horizontal qubit in the center, shown in red and numbered 1, with its odd coupler and paired qubit also in red, is internally coupled to vertical qubits, in pairs 3 through 8, each pair and its odd coupler shown in a different color, and externally coupled to horizontal qubits 2 and 9, each shown in a different color[9].

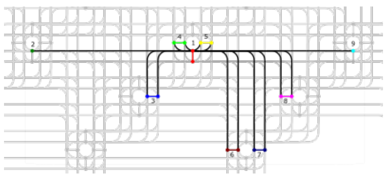


Figure 2.10. Coupled qubits "roadway" graphic (qubits represented as dots and couplers as lines): the qubit in the upper center, shown in red and numbered 1, is oddly coupled to the (red) qubit shown directly below it, internally coupled to vertical qubits, in pairs 3 through 8, each pair and its odd coupler shown in a different color, and externally coupled to horizontal qubits 2 and 9, each shown in a different color[9].

Pegasus Couplers

Internal couplers Internal couplers[9] connect pairs of orthogonal (with opposite orientation) qubits as shown in Figure 2.9. Each qubit is connected via internal coupling to 12 other qubits.

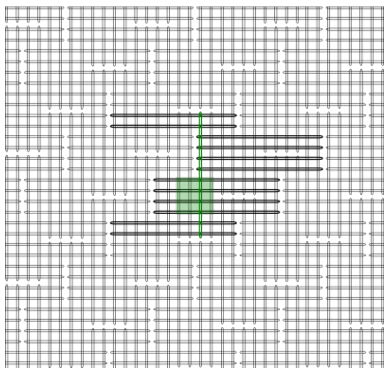


Figure 2.11. Junctions of horizontal and vertical loops signify internal couplers; for example, the green vertical qubit is coupled to 12 horizontal qubits, shown bolded. The translucent green square represents a Chimera unit cell structure (a K4,4 bipartite graph of internal couplings)[9].

External couplers External couplers[9] connect vertical qubits to adjacent vertical qubits and horizontal qubits to adjacent horizontal qubits as shown in Figure 2.13.

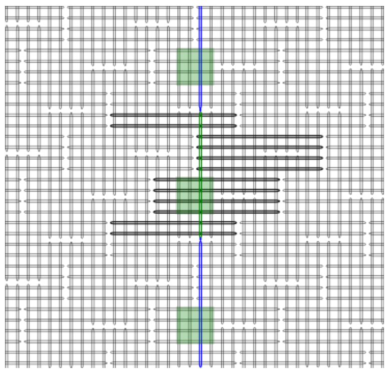


Figure 2.12. External couplers connect similarly aligned adjacent qubits; for example, the green vertical qubit is coupled to the two adjacent vertical qubits, highlighted in blue[9].

Odd couplers Odd couplers[9] connect similarly aligned pairs of qubits as shown in Figure 2.14.

Pegasus features[9] qubits of degree 15 and native K4 and K6,6 subgraphs. Pegasus qubits are considered to have a nominal length of 12 (each qubit is connected to 12 orthogonal qubits through internal couplers) and degree of 15 (each qubit is coupled to 15 different qubits).

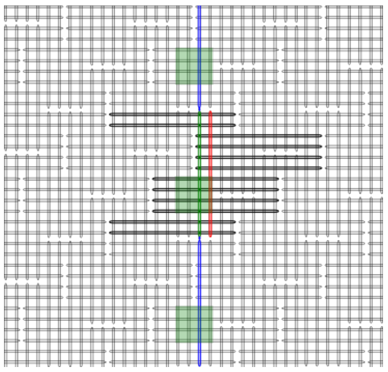


Figure 2.13. Odd couplers connect similarly aligned pairs of qubits; for example, the green vertical qubit is coupled to the red vertical qubit by an odd coupler[9].

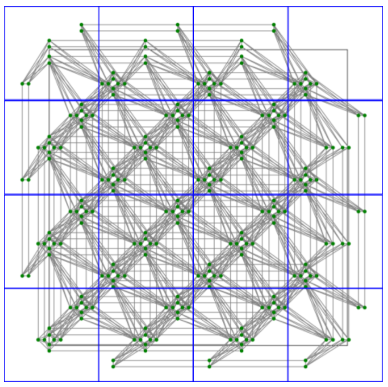


Figure 2.14. Pegasus unit cells in a P4 graph, with qubits represented as green dots and couplers as gray lines[9].

As the notation Cn refers to a Chimera graph with size parameter N, Pn refers to instances of Pegasus topologies; for example, P3 is a graph with 144 nodes. A Pegasus unit cell contains twenty-four qubits, with each qubit coupled to one similarly aligned qubit in the cell and two similarly aligned qubits in adjacent cells, as shown in Figure 22. An Advantage QPU is a lattice of 16x16 such unit cells, denoted as a P16 Pegasus graph.

More formally, the Pegasus unit cell consists of 48 halves of qubits that are divided between adjacent such unit cells, as shown in Figure 2.15.

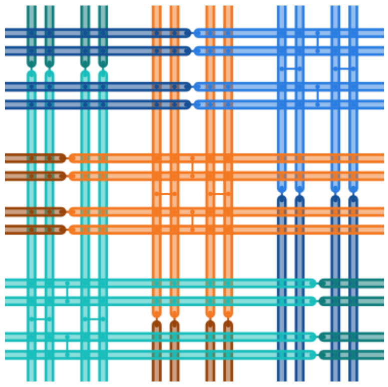


Figure 2.15. Pegasus unit cell shown as 48 halves of qubits from adjacent unit cells, with qubits represented as truncated loops (double lines), internal couplers as dots, and external and odd couplers as dots connected by short lines[9].

Chapter 3

Design of the solution

3.1 City graph

To evaluate the objective function and malus values a data structure that represents the city in question is necessary. This pool of information is provided by the Open Street Maps APIs, that return a graph in which nodes represent streets intersections and the edges represent the streets themselves.

For each node is available the latitude and the longitude, which is useful to extract pollution values using the BreezoMeter APIs, while for each edge is available the length of the street in question, useful to evaluate proportional weights for problem decision variables. Further, it is possible to save additional custom properties defined by the developer.

Open Street Maps has tools to print the city graph and this allows us to have a visual effect of the city and, more importantly, to better understand the solution found by the quantum optimization.

3.1.1 Open Street Maps

OpenStreetMap[5] is a collaborative project to create a free editable geographic database of the world. The geodata underlying the maps is considered the primary output of the project. The creation and growth of OSM has been motivated by restrictions on use or availability of map data across much of the world, and the advent of inexpensive portable satellite navigation devices.

Created by Steve Coast[5] in the UK in 2004, it was inspired by the success of Wikipedia and the predominance of proprietary map data in the UK and elsewhere. Since then, it has grown to over two million registered users. Users may collect data using manual survey, GPS devices, aerial photography, and other free sources, or use their own local knowledge of the area. This crowdsourced data is then made available under the Open Database License. The site is supported by the OpenStreetMap Foundation, a non-profit organisation registered in England and Wales.

For our aim it is used the OSMnx python library that allows with its APIs to download GeoSpatial data from Open Street Map and organize them in a graph structure. Are made available all the routine able to query and modify the graph, to allow the developer to have the full control or almost of the graph data structure.

OSMnx

OSMnx[6] is a Python package that lets you download geospatial data from OpenStreetMap and model, project, visualize, and analyze real-world street networks and any other geospatial geometries. You can download and model walkable, drivable, or bikeable urban networks with a single line of Python code then easily analyze and visualize them. You can just as easily download and work with other infrastructure types, amenities/points of interest, building footprints, elevation data, street bearings/orientations, and speed/travel time.

OSMnx[7] is built on top of GeoPandas, NetworkX, and matplotlib and interacts with OpenStreetMap's APIs to:

- Download and model street networks or other networked infrastructure anywhere in the world with a single line of code.
- Download drivable, walkable, bikeable, or all street networks.
- Impute missing speeds and calculate graph edge travel times.
- Simplify and correct the network's topology to clean-up nodes and consolidate intersections.
- Calculate and visualize street bearings and orientations.
- Visualize street networks as a static map or interactive Leaflet web map.

In the above list are only reported the features we are interested in.

3.2 Paths selection

Since the purpose of the algorithm is to select a path among a finite set, it is necessary to generate it. To better determine the best route solution it is fundamental that paths in the pool are quite different such that differences are not negligible, but similar enough to not generate an obvious best route. To fulfill the requirement, the number of paths required for the car is generated four times and subsequently is picked the pool path of the car. Selection is performed using a greedy algorithm, that represents a good compromise between quality of the solution and computation time. It was decided to use this algorithm after an analysis among different proposals which are described below.

3.2.1 K Shortest paths

Extraction of the larger pool is performed using the Open Street Maps API, in particular k_shortest_paths. For each pair of paths is evaluated the Jaccard similarity index, that is nothing more than the ratio between the number of common street segments and the union of segments sets. To better manage data they are organized in a symmetrical matrix the diagonal of which is obviously filled by 1 values since any path is equal to itself. The generated matrix is used to extract the main paths pool among which the quantum optimizer will select the best route. Results obtained using different algorithm described below, all take in consideration the real generated matrix showed in Table 3.2.2.1.

1.00	0.935	0.967	0.906	0.935	0.875	0.937	0.600	0.560	0.906	0.848	0.909
0.935	1.00	0.906	0.967	0.875	0.935	0.878	0.560	0.600	0.848	0.906	0.852
0.967	0.906	1.00	0.937	0.906	0.848	0.909	0.585	0.547	0.937	0.878	0.939
0.906	0.967	0.937	1.00	0.848	0.906	0.852	0.547	0.585	0.878	0.937	0.882
0.935	0.875	0.906	0.848	1.00	0.935	0.878	0.560	0.523	0.967	0.906	0.852
0.875	0.935	0.848	0.906	0.935	1.00	0.823	0.523	0.560	0.906	0.967	0.800
0.937	0.878	0.909	0.852	0.878	0.823	1.00	0.571	0.534	0.852	0.800	0.969
0.600	0.560	0.585	0.547	0.560	0.523	0.571	1.00	0.942	0.547	0.511	0.558
0.560	0.600	0.547	0.585	0.523	0.560	0.534	0.942	1.00	0.511	0.547	0.522
0.906	0.848	0.937	0.878	0.967	0.906	0.852	0.547	0.511	1.00	0.937	0.882
0.848	0.906	0.878	0.937	0.906	0.967	0.800	0.511	0.547	0.937	1.00	0.828
0.909	0.852	0.939	0.882	0.852	0.800	0.969	0.558	0.522	0.882	0.828	1.00

Table 3.1. Symmetric matrix that contains the Jaccard index similarity for each pair of paths

Combinatorial algorithm

This algorithm type provides the best possible solution in the face of a lack of computation time efficiency. Since the number M of paths for each car is known, it is necessary to build a combinatorial algorithm that creates a set of M elements among 4*M samples without repetitions. For each generated set the sum of mutual similarity Jaccard index is calculated and saved. Once all possible solutions are considered, the set that generates the smallest sum value is picked. Using data present in Table 3.1 we have a computation time equal to 2 milliseconds and an objective function equal to 1.88. Here is reported the code:

```
def rec_Combination(vector, numTot, numPaths, result,
    pos):
    global current_jaccard_value
    global matrix
    global best_result

    for i in range(numTot - numPaths + 1):
        result[pos] = vector[i]

        if(numPaths > 1):
```

```
        rec_Combinazioni([vector[j] for j in range(i +
           1, numTot)], numTot - i - 1, numPaths - 1,
           result, pos + 1)
    else:
        #One of the set is built
        if current_jaccard_value == -1:
            #evaluate value
            current_jaccard_value =
                matrix[result[0]][result[1]] +
                matrix[result[1]][result[2]] +
                matrix[result[0]][result[2]]

            #update best vector
            best_result = result

        elif (matrice[result[0]][result[1]] +
           matrix[result[1]][result[2]] +
           matrix[result[0]][result[2]]) <
           current_jaccard_value:
            #evaluate value
            current_jaccard_value =
                matrix[result[0]][result[1]] +
                matrix[result[1]][result[2]] +
                matrix[result[0]][result[2]]

            #update best vector
            best_result = result
```

Linear Convex Optimization

The problem is defined as an optimization problem and subsequently it is made linear and convex. The problem is composed by 4*M binary variables where M is the known number of routes of the car. This optimization problem is defined as follows:

$$\sum_{i=1}^{i=4*M-1} \sum_{j=i+1}^{j=4*M} var_i * var_j * matrix[i][j]$$

s.t.

$$\sum_{i=1}^{i=M} var_i = M$$

As it is formulated, it is neither linear nor convex, so a problem transformation is performed. This creates an objective function that is a relaxation of the previously defined one. In particular, variables are no longer binary but integer and they are constrained in a bound [0, 1]. Obtaining:

$$\sum_{i=1}^{i=4*M-1} \sum_{j=i+1}^{j=4*M} (var_i + var_j) * matrix[i][j]$$

s.t.

$$\sum_{i=1}^{i=M} var_i = M$$

$$var_i is integer; i = 1, ..., 4 * M$$

$$var_i <= 1; i = 1, ..., 4 * M$$

$$var_i >= 0; i = 1, ..., 4 * M$$

A mixed-integer problem solver provides the M best paths thanks to additional constraints. Using data present in Table 3.2.2.1 we have a computation time equal to 159 milliseconds and an objective function equal to 2.02. The obtained results are worse in terms of time and total Jaccard value, due to high number of decision variables and relaxations of the objective function.

Here is reported the code:

```
#Build optimization function
opt_f = 0
for i in range(numTot):
    for j in range(i + 1, numTot):
        opt_f += (decision[i] + decision[j]) * matrix[i][j]

#build the problem
problem = cvxpy.Problem(cvxpy.Minimize(opt_f), constraint)

#start time
start_time = time.time()

#solve the problem
problem.solve(solver=cvxpy.GLPK_MI)

#stop time
end_time = time.time()

#print result
print(sorted(zip(decision.value, list(range(numTot))),
   reverse=True)[:3])

#print time
print("Time: ", round(end_time - start_time, 4))
```

Similar paths merging

Another solution, but not valid at all, is to perform a merge of the most similar paths. This process must be repeated 3*M times to obtain the M paths we are looking for from a set of 4*M samples. Once that the similarity Jaccard matrix is evaluated the highest value, excluding the diagonal, is picked. Row and column represent the index of the two most similar paths and they can be aggregated. It is then needed to choose which path must be discarded, and since there are no immediate good choices, it is chosen at random. The matrix must be updated and the deleted path must be taken into account to correctly extract the paths pool. Due to data rearrangement and picking the path at random, this option was discarded apriori without evaluating the performance.

Greedy algorithm

Last but not least, it is presented the greedy algorithm that is actually the chosen method to pick the paths pool. This is a native methodology that associates at each path a Jaccard similarity index. This is simply obtained by averaging the similarity values of other paths. At this point the M lowest values are picked that, obviously, represent the paths pool. Using data present in Table 3.2.2.1 we have a computation time in the order of microseconds and an objective function equal to 2.02. It is possible to immediately see that computation time is one or two orders of magnitude smaller than the one of other algorithms and the quality of the solution is quite near to the optimum. In general the paths pools share 66% of the street segments, therefore fully satisfying the initial requirements.

Here is reported the code:

```
def ExtractMostDifferent(paths, numberOfRoutes):
    meanJaccard = [0]*len(paths);

    #calculate mean jaccard values
    for i in range(len(paths)):
        for j in range(i, len(paths)):
            if i =! j:
                jaccardValue = JaccardIndex(paths[i],
                   paths[j])
                meanJaccard[i] += jaccardValue
                meanJaccard[j] += jaccardValue

    indexes = [tuple[1] for tuple in
       sorted(zip(meanJaccard,
       list(range(len(paths)))))[:numberOfRoutes]]

    resultList = []
    for index in indexes:
        resultList.append(TransformPath(paths[index]))

    return resultList
```

3.3 QUBO matrix

Starting from the QUBO matrix built in section (inserisci la sezione dell'algoritmo di volkswagen), we will perform a perturbation in order to consider in the evaluation of the optimum the coexistence of the car along a path and the pollution existing on the path itself as well as the one produced by cars. These two parameters are not unrelated since the greater the coexistence time is, the larger will be the pollution produced by each car in a traffic jam. Obviously, hybrid and electric cars will have a reduced or null impact in this scenario. The main reason why these two parameters are taken in consideration is because we are trying to minimize the air pollution during usage of single transport vehicles for the arguments treated during the global pollution section.

3.3.1 Coexistence penalties evaluations

Time spent in the traffic jam is a source of air and noise pollution that, willing or not, brings to a lower quality of life. The aim of this section is to evaluate in a significant way the values associated with the environmental damage that can be caused by this kind of scenario. Therefore, there is the need to split the traffic jam to avoid traffic congestion. It's obvious the immediate improvement relative to the actual situation, analogous behavior can be encountered in internet network while performing packet traffic engineering. The developed algorithm is able to evaluate the number of delta times that two different cars spent on the same street segment, assuming that velocity is constant for each vehicle and without interrupting the motion. To better simulate the reality, different release times can be defined since it is implausible that all trips start at the same time. Moving to algorithm logic implementation, each path is represented as a sorted list of street segments, that remembering the graph structure, are composed by two graph nodes. From Open Street Maps it is possible to extract the length of each street segment and dividing it in equal sub-intervals it is possible to simulate the passage time. At each delta a check is performed on which street segment a car is present on, if the segment results to be a common one a counter is incremented to keep track of the number of common delta time occurrences considering two specific cars and paths. This operation is repeated for each couple of cars and paths, to correctly evaluate every coexistence delta and consequently perturb the QUBO matrix in its entirety. This process is repeated of each pair of paths belonging to two different cars, then the calculated value is scaled to be consistent with values already present and inserted in the correct matrix cell. The proportional value is an empiric one and at the moment is set to 10, this works well in a little city with a smaller number of cars and paths. If the algorithm is run on a large-scale problem, it is necessary to reduce it in order to not overshadow the basic QUBO values and cause a slightly imbalanced result.

This is the core function that evaluates the common delta time for each paths pair:

```
def EvaluateCohexistanceTime(QUBOdict, graph, routesPerCar,
    numberOfCars, realiseTimes, routesDict):
    #instantiate result
    result = {}

    #invoke cycle evaluate each paths pairs among different cars
```

```
for looperCar1 in range(numberOfCars - 1):
    for looperCar2 in range(looperCar1 + 1, numberOfCars):
        #for each route of the first car, check the route of
            other car
        for routeNum1, routeCar1 in
            enumerate(routesPerCar[looperCar1]):
            for routeNum2, routeCar2 in
                enumerate(routesPerCar[looperCar2]):
                #initialize data
                currentIndex = [0] * 2
                currentSegment = [(0, 0)] * 2
                segmentLength = [0] * 2
                currentDistance = [0] * 2
                currentTime = 0
                cohexistanceTime = 0

                allActive = False
                loop = True
                while loop:
                    if not allActive:
                        allActive = all(segment != (0, 0) for
                            segment in currentSegment)
                    else:
                        loop = all(segment != (0, 0) for
                            segment in currentSegment)


                    #get corrispective segment for each route
                    for car in range(2):
                        if car == 0:
                            routeCar = routeCar1
                            timeIndex = looperCar1
                        else:
                            routeCar = routeCar2
                            timeIndex = looperCar2

                        #check realise time
                        if(currentTime >=
                            realiseTimes[timeIndex]):
                            if(currentIndex[car] <
                                len(routeCar)):
                                currentSegment[car] =
                                    routeCar[currentIndex[car]]
                                segmentLength[car] =
                                    graph.get_edge_data(currentSegment[car][0
                                    currentSegment[car][1])[0]['length']
                            else:
                                currentSegment[car] = (0, 0)
                                segmentLength[car] = 0

                    #check if the car are in the same segment
                    if (all(segment != (0, 0) for segment in
                        currentSegment)) and currentSegment[0]
                        == currentSegment[1]:
                        cohexistanceTime += 1

                    #update time and if necessary the segment
                    currentTime += 1

                    for car in range(len(currentDistance)):
                        if(currentSegment[car] != (0, 0)):
                            currentDistance[car] += 1

                            if(currentDistance[car] >=
                                segmentLength[car]):
                                currentDistance[car] -=
                                    segmentLength[car]
                                currentIndex[car] += 1

                #save cohexistance time of the routes
                result[((looperCar1, looperCar2), routeNum1,
                    routeNum2)] = cohexistanceTime

#evaluate dict data
CohexDict = EvaluateWeight(graph, routesPerCar, result)
```

Once common delta times are evaluated, another native routine, EvaluateWeight, calculates the effective malus values that must still be scaled and puts all values in a dictionary the keys of which are composed this way:

$$((carNumber_0, carNumber_1), i - thPath_carNumber_0, i - thPath_carNumber_1)$$

Where:

- $\mathrm{carNumber}_0$ is the index of the first considered car
- $\mathrm{carNumber}_1$ is the index of the other considered car
- $\mathrm{i\text{-}thPath}_{\mathrm{carNumber}_0}$ is the i-th path of the first car
- $\mathrm{i\text{-}thPath}_{\mathrm{carNumber}_1}$ is the i-th path of the other car

The final coexistence value is a proportional normalization of the paths pair. At first the value is normalized relative to half the sum of the length of each street segment of both paths and subsequently multiplied by the minimum value between the number of segments of the two paths. This way it is maintained the Volkswagen common segment proportionality in order to not distort the matrix values and optimization function behavior. To better clarify the concept is reported the code:

```
def EvaluateWeight(graph, routesPerCar, cohexistanceDict):
    dictResult = {}

    #for each key do something
    for key in cohexistanceDict:
        value = cohexistanceDict[key]

        #get route of first car
        route1 = routesPerCar[key[0][0]][key[1]]

        #get route of second car
        route2 = routesPerCar[key[0][1]][key[2]]

        #evaluate the total length
        totLength = 0

        for segment in route1:
            totLength += graph.get_edge_data(segment[0],
                segment[1])[0]['length']

        for segment in route2:
            totLength += graph.get_edge_data(segment[0],
                segment[1])[0]['length']

        #evaluate percentage
        percentage = 2*value/totLength

        #get the number of segment of the shortest
        totSegment = min(len(route1), len(route2))

        #add value to new dict
        dictResult[key] = 2 * totSegment * percentage

    return dictResult
```

The routine get_edge_data() is an Open Street Maps APIs that extracts an edge data structure given the extreme nodes. This object contains edge features including the length of the segment.

As mentioned before dict values as inserted in the correct QUBO cell with the empirical scale factor equal to 10:

```
#perturb QUBO dict
totalRoutes = sum([len(carRoutes) for carRoutes in routesPerCar])
reductionFactor = 10

for key in CohexDict:
    car1 = key[0][0]
    car2 = key[0][1]
```

```
route1 = sum([len(routesPerCar[i]) for i in range(car1)]) +
   key[1]
route2 = sum([len(routesPerCar[i]) for i in range(car2)]) +
   key[2]

row = routesDict[route1]*totalRoutes + route1
column = routesDict[route2]*totalRoutes + route2
if (row, column) in QUBOdict:
    QUBOdict[(row, column)] += reductionFactor *
       CohexDict[key]
else:
    QUBOdict[(row, column)] = reductionFactor * CohexDict[key]
```

There are three main improvements that can be performed:

- Consider an heterogeneous car's velocity along each single street segment: once it is known the speed limit of a specific segment, it can be better to take a safer velocity, for example the 90% of the speed limit and tune the delta times slot based on this information.
- Consider car stops during the trip: In a real situation, especially in a city, there are traffic lights and crossroads that can slow down the march. Knowing their position it can be evaluated a probabilistic percentage that a specific car velocity slows down or completely stops the motion. These probabilistic values can be evaluated based on the number of roadways in the street segment, the length and mean times that a traffic light stays green considering the characteristic of the street. It is not necessary that is entirely accurate since reality can't be reproduced identically, however one or more non-deterministic events can occur that can change the evaluated percentage.
- Coexistence delta doesn't take in consideration the length of the street: Actual implementation considers only if two cars are located on the same street segment at a given time, not taking in consideration the distance between cars. In a certain way this is not relevant at all, because in a city the distance between crossroads is not enough to make a consideration of this type significant.

3.3.2 Pollutions penalties evaluation

The evaluation of the values belonging to this section is the real core argument of this work. It is necessary to calculate and scale as well as possible the malus brought by street and cars pollution. Sensors scattered around the city are able to measure the PM10 and PM25 pollution level with sufficient accuracy and every significant time slot. These levels, obviously, depend on the number of cars that pass through the street, air circulation, conformation of the road, the weather and time slot. It is clear that a path possesses intrinsically a pollution cost that must be taken in consideration. Therefore, QUBO matrix will be perturbed in two ways:

- On the diagonal: A value is summed on the main matrix diagonal to take in consideration the intrinsic cost of full path.

On cross cells: A value that takes in consideration the pollution cost of the two cars on the same street for each common segment. To evaluate the full path pollution cost it is applied this formula:

$$\sum_{s=1}^{s=n}(PMvalue(s_0) + PMvalue(s_1)) * length(s)/2$$

Where:

- s is the s street segment of the path
- s_i the first node of the segment
- PMvalue is the routine that extracts the PM10 and PM25 values at current time in precise coordinates of the earth. It is used BreezoMeter APIs to retrieve these values.
- length is the routine that evaluates the length of the segment, It can be retrieved by a property of the Open Street Maps graph edges.

To the resulting value is applied a scale factor equal to 0.001 and added to the diagonal of the matrix for the corresponding path. For what concerns the pairs between paths belonging to different cars, the formula is applied only for common segments and not for all the length of roads. Since this value in multiple runs seems to be smaller, it is applied a factor of 0.01.

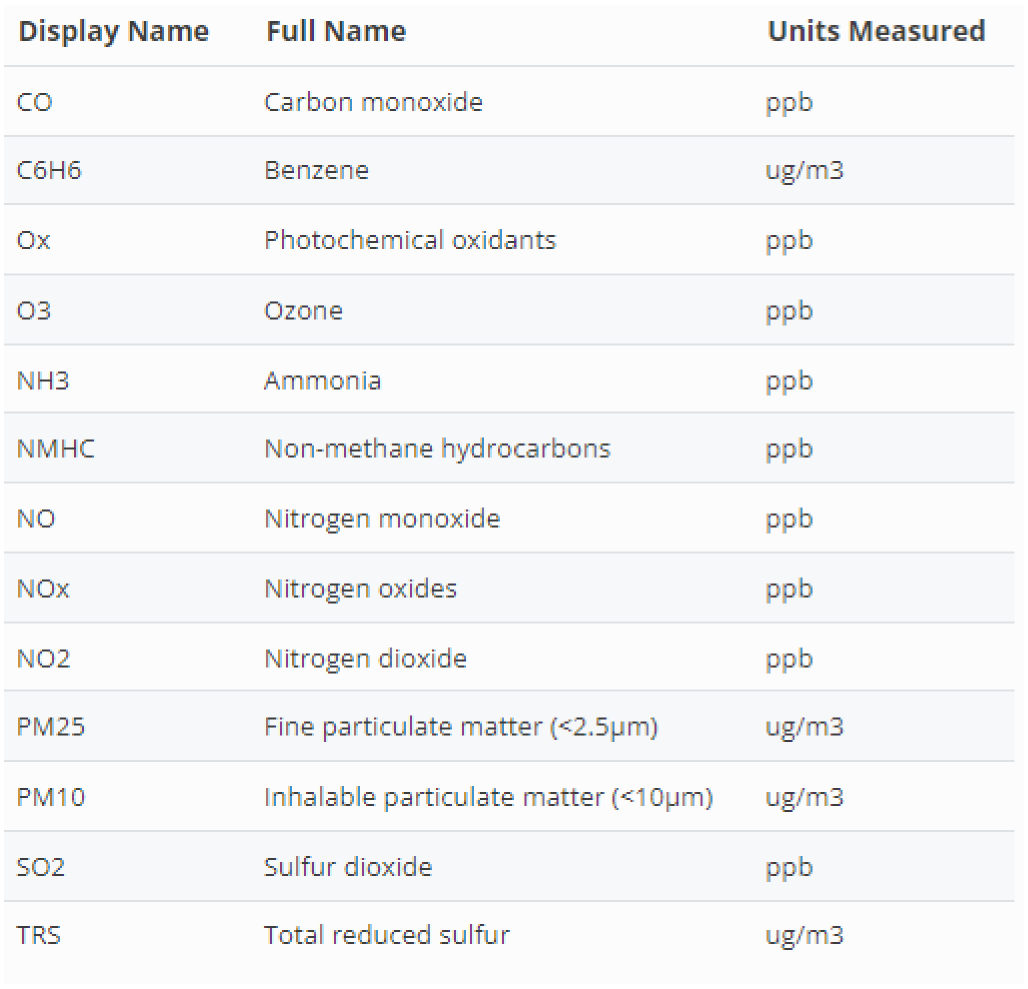
To complete the enhancement of the objective function, just one factor is missing, that is the one provided by car topology. At the moment only two types of car are defined:

- Petrol Car: Pollution cost is a non zero value, since petrol cars produce pollutants even if the car is not going. This is because the engine continues to operate and needs to consume petrol, it is a reduced form of pollution, but not negligible.
- Electrical Car: pollution cost for those car is zero.

These factors are taken in consideration only when the car is not in motion, but is turned on. Considering the simplification we performed in the previous section, cars don't stop during the trip, objective function has not additional intake. It is possible to define new car types defining the related cost function. In this way, it is possible to fine tune the model and have, as far as possible, the best path selection.

BreezoMeter

BreezoMeter[4] is a company where scientists, engineers and professionals try to improve the health and quality of life for billions of people worldwide, by providing the world's most accurate and actionable environmental data and insights. They provide[4] data about air quality, pollen level, wildfire tracker and weather all in the form of APIs. Naturally the one we are interested on is the air quality APIs that is able to provide not only data relatively to PM10 and PM25, but also for:



Display Name	Full Name	Units Measured
CO	Carbon monoxide	ppb
C6H6	Benzene	ug/m3
Ox	Photochemical oxidants	ppb
O3	Ozone	ppb
NH3	Ammonia	ppb
NMHC	Non-methane hydrocarbons	ppb
NO	Nitrogen monoxide	ppb
NOx	Nitrogen oxides	ppb
NO2	Nitrogen dioxide	ppb
PM25	Fine particulate matter (<2.5µm)	ug/m3
PM10	Inhalable particulate matter (<10µm)	ug/m3
SO2	Sulfur dioxide	ppb
TRS	Total reduced sulfur	ug/m3

BreezoMeter pollutants table[4]

Pollutants levels[4] are updated hourly that result to be a good time slot, since levels don't change in a couple of minutes. It is also possible to retrieve some additional and useful informations about pollutants level health risk and a bunch of prevention measure. In conclusion, BreezoMeter represent a good solution to retrieve accurate and update information in all over the world just inserting the latitude and longitude of desired point.

BreezoMeter cache

To avoid multiple and useless BreezorMeter APIs calls, it was natively implemented a cache using the structure of a json file. It is composed by a list of Json object which has the following structure:

```
NodeNumber: {
        "PM10": double,
        "PM25": double,
        "SampleTime": "hh:mm:ss"
},
```

Where:

- NodeNumber: it is a long int that uniquely identify a node of the graph.
- PM10 and PM25: Pollutants values level at that node position.
- SampleTime: indicate the time when the BreezoMeter's API call was performed.

When it is necessary to know pollutants level of a given node, it is first checked if the necessary data is already present in the cache and also that it is not older than an hour. If not, an API call is performed, and the cache is updated. This way if the algorithm is run more than once in one hour, if the requested node is already present, we have a performance increase.

The algorithm is really simple and it is reported below:

```
#get cache file of a specific city
cacheFileName = city + 'cacheData.json'
cache = {}
try:
    file = open(cacheFileName)
    cache = json.load(file)
    file.close()
except FileNotFoundError:
    pass

#flag variable to update cache file
cacheUpdate = False

#instantiate brezometer
pollution = apiPollution()

#instantiate data
diag = 0
reductionFactor = 0.001
routePollutions = []

#cycle over node
for car, routes in enumerate(routesPerCar):
    for routeIndex, route in enumerate(routes):
        #generate a set from the node list
        node_list = []
        node_list.append(route[0][0])
        node_list.extend([nodes[1] for nodes in route])

        pathTotalPollution_PM25 = 0
        pathTotalPollution_PM10 = 0

        for node in node_list:
            #check if data is in the cache
            if node not in cache or
               (datetime.strptime(cache[node]['SampleTime'],
               '%H:%M:%S') - datetime.now()).total_seconds() >
               3600:
                #retrieve data and add it to cache
                nodeData = {}

                #retrieve node
                Gnode = G.nodes[node]
                pollution.queryPM(Gnode['y'], Gnode['x'])

                nodeData['PM10'] = pollution.pm10_now
                nodeData['PM25'] = pollution.pm25_now
                nodeData['SampleTime'] =
                   datetime.now().strftime("%H:%M:%S")
                cache[node] = nodeData
                cacheUpdate = True
```

apiPollution is a class that contains all BreezoMeter APIs call logic, it collects all necessary data to perform a request and manage the result.

3.4 Quantum solving

Now we approach the most interesting section of the work: Resolution of the problem using D-Wave QPU. Initially is needed to build the quantum solver, that is a really simple operation thank to intuitive APIs provided by Leaf service.

```
solver =
   EmbeddingComposite(DWaveSampler(solver={'topology__type':
   'chimera'}))
```

It was chosen a chimera architecture for QPU since it has parameters that Pegasus one doesn't have. In particular, it was of particular help the parameter linked to the Boltzmann constant, since after the problem is solver, it change the way in which solution is picked. As in statistical mechanics, β represents inverse temperature: 1/(kB*T), where T is the thermodynamic temperature in kelvin and kB is Boltzmann's constant. In the D-Wave software, postprocessing refines the returned solutions to target a Boltzmann distribution characterized by β, which is represented by a floating point number without units. From documentation for lower β values, sampling is less constrained to the lowest energy states. So it was tuned performing several run since a good compromise is found between the number of cars and the number of picked paths.

Additional parameters are take in consideration, such:

- Annealing time: Sets the duration, in microseconds with a resolution of 0.01 µs for Advantage and 0.02 µs for D-Wave 2000Q systems, of quantum annealing time, per read. This value populates the qpu_anneal_time_per_sample field returned in the timing structure. Supported values are positive floating-point numbers.
- Number of reads: Indicates the number of states (output solutions) to read from the solver. Must be a positive integer.

Once that solver is built and parameters are sets, it is necessary to invoke the routines that perform the solving. The method in object is sample_ising() that accept a problem in Ising form and perform the optimization. Result is a dict that has length equal to the number of decision variables and value 1 or -1 respectively if the current variable is active or not. Since parameters tuning is performed in an empirical way, there is not the certainty that result is consistent. Therefore, a check on the result is performed: if the active variables are not equals to the number of cars or was picked more than one path for a single car, the solution is discarded and the subsequent, that has an higher energy level is picked and tested.

3.4.1 Ising problem

The Ising model[2] is a particular example of a thermodynamic system, it is particularly interesting because it's one of the few models that are exactly solvable and provide thermodynamic quantities that we are able to interpret. The ising model is defined as a "mathematical model that doesn't correspond to an actual physical system. It's a huge lattice of sites, where each site can be in one of two states. We label each site with an index i, and we call the two states -1 and +1", from this definition it is clear the similarity with the QUBO matrix problem formulation. For instance it is considered a model for "magnets" to which it is possible to associate an energy value, useful to set the objective function and the penalties as discussed in previous sections. This models has two types of interaction:

- External field is an external magnetic field that is able to modify the energy values associated with each state. This principle is the one that is used to perturb the QUBO matrix in the quantum computer.
- Interaction between qubits that react to each other (Entanglement) to bring the system in a specific pool of states. It is necessary to highlight pool because the solution evaluation happens in parallel, so multiple are returned. Note that each solution has a different value of total energy.

The Hamiltonian of the Ising Model can be written as:

$$H = -\sum_{<i,j>} J\sigma_i\sigma_j - \sum_j h\sigma_j$$

The two sum represent the two values that it is necessary to provide to D-wave APIs to correctly build the constant matrix system in the quantum computer.

3.5 Algorithm structure

The written algorithm presents a unique entry point, since once the input data is received, the quantum implementation logic is treated as a black box, from which are returned the selected solution paths, one for each car. The input data is represented by QuantumInput classes, that are passed as argument to the static function "QUBOimplementation()", that is the only routine visible from outside.

3.5.1 Data structure

The input data, "QuantumInput", is built as a class, which contains five fields:

- NumberOfCars: It is an int value that represents the number of cars of which algorithm must perform the optimization.
- NumberOfRoutes: It is a list on int values, that contains NumberOfCars items and represents the size of the paths pool for each car.
- RealiseTimes: It is a list of int values, that contains NumberOfCars item and represent at which delta time a car start its trip.
- CarTypes: It is a list of enum values, that contains NumberOfCars items and represents the types of the cars.
- SourceTargetList: it is a list of tuple values, that contains NumberOfCars items. Each tuple contains two objects that respectively represent the starting and the ending points.

3.5.2 QuantumSection composition

Algorithm of the new quantum section is divide in four macro blocks:

- QuantumSection: it is the main block, it is in charge of calling the others routines, to invoke the quantum solver and to print the results.
- QUBObuilder: Generate the paths pool for each cars and build the starting QUBO matrix provided from previous Volkswagen study.
- RoutesCoexistence: Place in which algorithm that evaluate the perturbation for cars coexistence time takes place.
- Pollutions: In addition to generate the values related to pollutants levels, manage the city cache presented in section "BreezoMeter cache".

I would like to bring attention on the first part of QUBObuilder block, since it is where is built the main structure use during problem evaluation. To facilitate the evaluation is create a list of lists, in particular the first order list own NumberOfCars lists and each list contains NumberOfRoutes path. In this way is immediate to understand which path belongs to which cars and in addition it is possible to fast pairs paths belonging to different car pools.

Chapter 4

Results

4.1 Correctness check of the algorithm

Before starting to discuss results, it is necessary to check the consistency and correctness of the algorithm. To reach this goal is take in consideration a small town, Cavallermaggiore, with more or less 5000 street intersections and three cars. To perform this check it was chosen two common points that represent the initial and final position of the cars, from which is generated a common paths poll containing three paths. The expected result is that the three cars take each a different path. In the first runs I noticed that sometimes we retrieve an invalid solution since a car had two selected routes and the other not one. So, I have acted on the λ value that determine the influence of hard constraints, discovering that the "correct" value is composed of two terms: λ itself and a scale factor that depends on the number of cars and the number of total routes.

$$Hardconstraints = \lambda * f(n, \sum_m numberOfRoutes)$$

$$f(n, \sum_m numberOfRoutes) = \sum_m numberOfRoutes/n$$

In this way we obtain an acceptable solution in which each car have selected a path from its pool. Thank to penalties definitions, each car take a different route among the common path pool. Here, are reported visually one of the solution that I obtain, in which figure 4.1 show the entire town, instead the figure 4.2 show a particular, specifically the common start location.

From the figure 4.2, it is possible to see the street differentiation, since the overlapping of the routes produces a more intense red color. Following the different branches it is possible to notice the intensity decreasing of the red color, in particular at first intersection two cars tuns left and the other turn right, subsequently, the other cars take different paths two intersection later.

Now that we have an evidence of the correctness of the written algorithm, we can begin to discuss the results, in which are reported some samples of the algorithm functionality, in addition to make a comparison between classic qbSolve and quantum annealing.

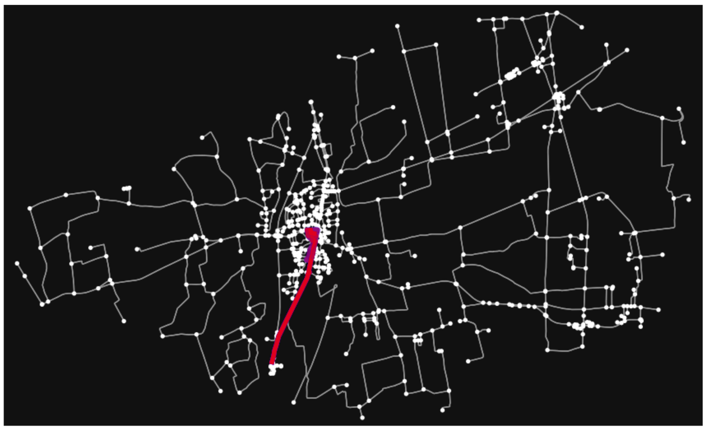


Figure 4.1. Full town solution

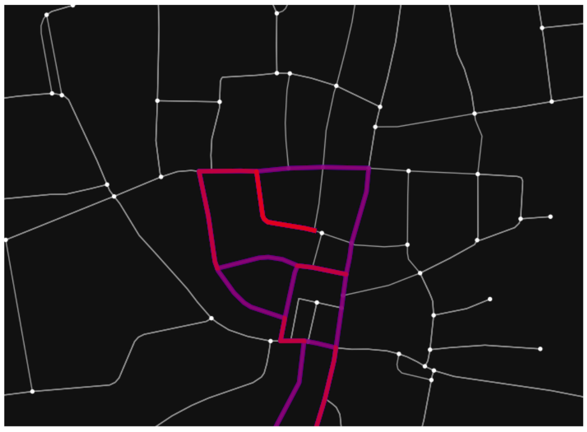


Figure 4.2. Particular of the staring point of the three cars

4.2 Result comparison between qbsolve and QPUs

I'm going to show the results obtained by my algorithm, considering two different kind of solver, the first is the classic qbsolve and the second is the D-wave quantum solver. Other than visual result report will be also provided the index of the path chosen by a car and the relative energy of that specific solution. It will be analyzed the difference at each passage, represented by:

- Basic formulation
- Basic formulation + cars coexistence
- Basic formulation + cars coexistence + environmental pollution

In order to explore all the scenario that this algorithm is able to solve, different city, number of cars, number of routes, release time and boundary conditions are take in considerations. For the subsequent run, for which concern the quantum computers, are used those values as QPUs parameters:

- Annealing time: 2
- Number of reads: 10
- beta: 100 (Boltzmann parameter)

The first two examples consider the city of Cavallermaggiore and the last one the city of Turin, and will be analyzed the solution change when additional penalties are considered. Considering a small number of cars, few number of routes as path pool or quite different values of release time, it may happen that coexistence penalties does not bring any contribution to the solution, subsequently the base formulation and the one that consider the coexistence time are going to be equal, in terms of chosen paths and, obviously in energy value.

4.2.1 First run

As other runs initial and final points are choose at random among the intersections of the city. Following are shown the input data of the algorithm:

- Number of cars: 3
- Number of routes: 3, 3, 3
- Release time: 0, 0, 0

Basic formulation:

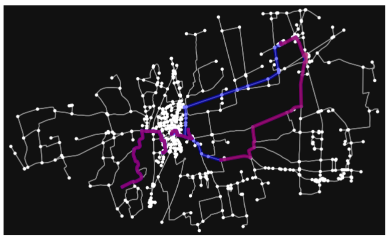


Figure 4.3. Qbsolve solver

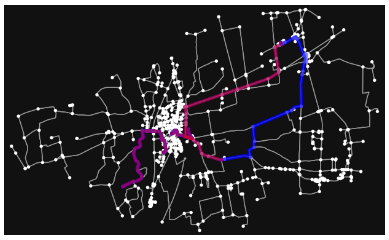


Figure 4.4. D-wave Chimera solver

Considering the figure 4.3 which represent the qb solve solution, the three cars take respectively the paths: 2, 1, 1; and have an energy of -544. Instead the figure

4.4 represent the D-wave QPUs solution in which the three cars pick respectively the paths: 2, 3, 3; and have an energy of -3028. Since the solver are different it is not possible to compare the energy levels, the reported values have a initial condition purpose. Adding new penalties we are going to see the differences.

Since the coexistence time has a zero value for each paths pairs, we immediately jump to analyze the impact that has the pollutions on the solutions.

Pullution formulation:

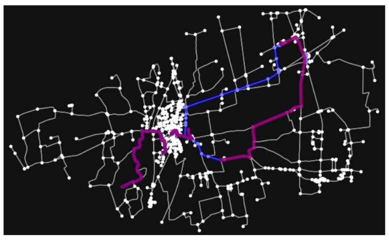


Figure 4.5. Qbsolve solver

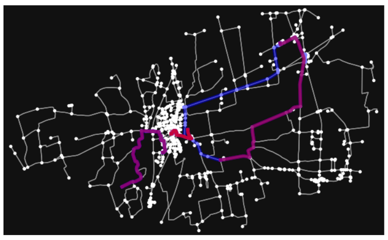


Figure 4.6. D-wave Chimera solver

Considering the figure 4.5 which represent the qb solve solution, the three cars take respectively the paths: 2, 1, 2; and have an energy of -374.27.
Instead the figure 4.4 represent the D-wave QPUs solution in which the three cars pick respectively the paths: 1, (1-3), 2; and have an energy of -2714.
Considering the solution provided by qb solve it is interesting that the last car change its path from the first to second, this means that pollution concentration is a lot more impactful, respect to the length of the path and geographical location. Since a longest path can be less polluted even considering the scale factor associated with the street segment length.
For which concern the quantum solver, the presence of the penalties linked to the pollution, heavily impact the previous found solution. As it shown all cars change the previously chosen path, except for the second one that unfortunately have picked two path as solution, this result can be explained in different ways, or better it can be the result of small uncertainties that put all together are no more negligible, such:

- Wrong or not well defined of the penalties parameters
- QPUs parameter are not good at all, they need to tuned better
- High read error from the point of view of the quantum hardware

As expected, the energy of the solutions in the second case are significantly larger than the first case, since there are present penalties that contribute to create an augmented objective function and consequently an highest energy value.

4.2.2 Coexistence run

Taking in consideration the same city, Cavallermaggiore, we are trying to force the presence of coexistence of the car using more number of routes for each cars, following are reported the input data of the algorithm:

- Number of cars: 3
- Number of routes: 5, 10, 7
- Release time: 0, 10, 5

Due to issues discussed earlier about quantum solving multiple solution, only qb solve is used to evaluate results, since quantum solving generate more than one solution for each car, even two different path has more or less the same energy considering both coexistence and pollution.

Basic formulation:

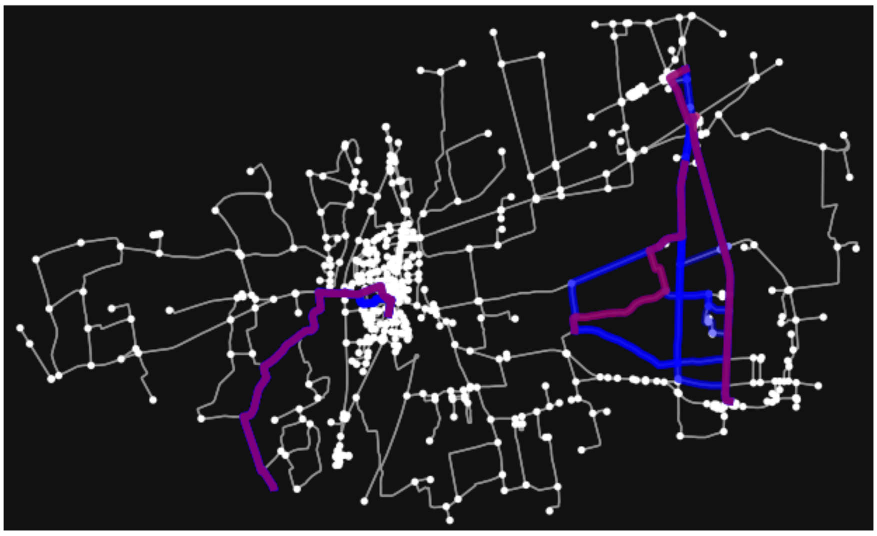


Figure 4.7. Qbsolve solver basic formulation

Considering the figure 4.7 the three cars take respectively the paths: 5, 5, 1; and have an energy of -25398. Respect to the previous run the considered path is more, this has a consequence on the values of the QUBO matrix that statistically can be largest than the previous run and this bring the solution to have an energy than differ of some magnitudes order.

Coexistence and Pollution formulation:

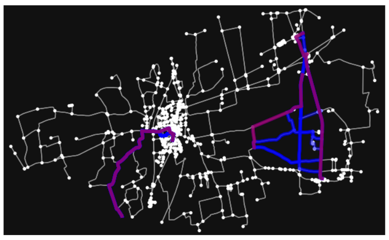


Figure 4.8. Qbsolve solver coexistence

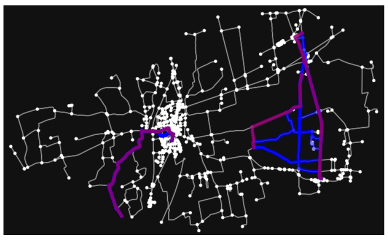


Figure 4.9. Qbsolve solver coexistence plus pollutions

Figure 4.8 is the solution retrieved adding the coexistence penalties, producing the following result: 5, 5, 2; and have an energy of -25327. Looking at the right part of the image is obvious that car two and three share a lot of segment, many of which present the two cars in the same time slots. Those penalties are obviously take in consideration by the solver and bring the third car to change its solution path in order to not congest the same street segment of the second car.

Figure 4.8 is the solution retrieved adding the pollutions penalties (also considering the previously coexistence penalties), producing the following result: 3, 5, 1; and have an energy of -25214. I think that this result is the most important of the research since highlights the importance of the penalties related to pollutions; focusing on second and third car, solution is brought back to the first one, since penalties connected to coexistence penalties are negligible (in this case) respect to the pollution one, the alternative 2 for the third car over extend the length of path to reach the destinations and this have a strong impact on the scale factor of the pollution levels. In this scenario it is better to congest a street segment respect to pollute more.

As the previous case adding penalties increase the values of the total energy, so we can consider the result consistence since the behavior is the one expected.

4.2.3 Turin run

It is also analyzed a biggest city, Turin, to emphasize the power of this algorithm, obviously are take in consideration a largest number of cars and different number of routes that constitute the paths pool of each car, those are the input data of the algorithm:

- Number of cars: 5

- Number of routes: 4, 3, 5, 2, 4
- Release time: 20, 5, 34, 27, 12

Basic formulation:

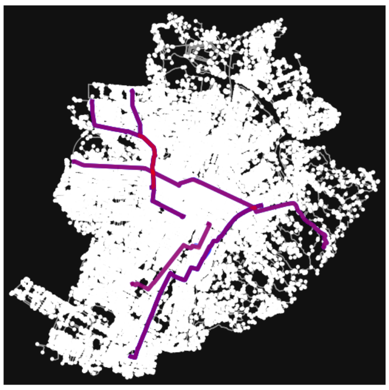


Figure 4.10. Qbsolve solver basic formulation

Considering the figure 4.10 the five cars take respectively the paths: 0, 3, 2, 2, 4; and have an energy of -28384. Respect to the previous run the considered cars and number of paths are more, this has a consequence on the values of the QUBO matrix that statistically can be largest than the previous run and this bring the solution to have an energy than differ of some magnitudes order.

Pollution formulation:

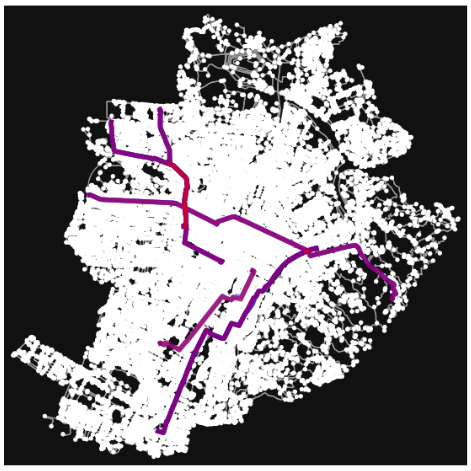


Figure 4.11. Qbsolve solver Pollution formulation

Figure 4.11 is the solution retrieved adding the pollution penalties (No car coexistence in this run), producing the following result: 0, 3, 2, 4, 4; and have an energy of -27067. As shown in the image, only the fourth car has some overlapping path and thanks to defined QUBO matrix it change its path to avoid polluting and traffic congestion.

Chapter 5

Conclusion

5.1 Final conclusion on the results

During the result analysis, were compared the results divided between solvers and incremental penalties, so we can make some considerations.

- Thanks to the increasing curve generated by energy values and the consideration on common street segment and length on figure, we can conclude that results are consistent.
- Energy comparison must be performed for the same solver, since energy levels are interpreted and evaluated in different ways even if the meaning is the same. We can have a confirmation taking a look on the section 4.2.1, energy levels differ of some magnitudes order and the picked paths are different.
- Quantum solver and classical solver for problem of this entity have comparable computation time in order to compute a solution, but it is important to highlights that for larger problem is crucial to use a quantum solver to drastically cut the searching time.
- Thanks to QUBO formulation the number of decision variables is quite small respect to other kind of formulation and for this reason solving can be performed also with classical solver.
- It is necessary to find a function that consider a bunch of the parameters of the problem to correctly tune the quantum parameters, in order to find a consistent solution regardless of the problem size. With my knowledge i found values for quantum parameters that are suitable for problem composed by no more than 3 cars and no more of 9 total routes.
- Solving process is quite fast, but the problem formulation require a significant amount of time (compared with computation time to find a solution), it is necessary to consider a lot of variables to correctly build the penalties related to the cars coexistence and environment pollution.

To conclude, I think that quantum solving is the way to go, but at the moment due to hardware limitation, poor knowledge on the quantum field and limited distribution of those powerful devices, it is an unripe technology that needs more and more improvements to be fully exploited. Having regards to the fact that performing a tuning for a specific king of problem, and setting aside the scalability, remains a powerful to speed up the decision-making process on a specify field.
However, this algorithm represent an initial solid solution to drastically reduce, first of all, the pollutants released in the environment produced by daily traffic jam all over the world and not less important to increase the traffic flow in the city performing a sort of traffic engineering as networker are trying to do with internet packages. It can be surely improved from the point of view of penalties building performance and number of covered scenario, but it is a solid milestone that can realistically improve the life quality of each person and why not, of all living creature and our planet Earth.

5.2 Possible future improvements

As we already mentioned in the previous sections there are many possible improvements that can be performed on the two new blocks of the algorithm, respectively coexistence and pollution evaluation.

5.2.1 Coexistence block improvements

At the moment to simplify the coexistence evaluation, velocity of the cars is considered constant at all: constant between cars and for each type of street segment, independently from street condition, viability and number of railways. To better simulate the reality it is necessary to take the speed limit of the segment in which the car is moving on and apply a safety mean margin on the maximum velocity, for example 90%. Further it is also possible to consider a % penalties based on the number of cars in the segment and the number of railways. In this way, we are trying to better simulate the real velocity on the segment and consequently evaluate with more precision the number of delta times that two or more cars coexist on the same street segment.

Another possible improvement is to simulate semaphore or the intersections in which the motion of a car stop. This parameter can be expressed in the form of % based on data collected about the mean car stopping and congestion in presence of blocking factor in function of size of intersection, green light duration if it is present and number of railways to simulate the street flows.

In general can be considered also all the factors that contribute to stop of slowdown the traffic jam. Fall into this category:

- Reaction time of the driver
- Possible presence of cars accident
- Street congestion at peak times

- Weather
- ...

5.2.2 Pollution block improvements

At the moments for QUBO penalties are only considered PM10 and PM25, it can be possible to considered also other pollutants always produced by engine combustion. Note that if more factor comes to perturb the QUBO matrix, it is necessary to tune again the associated scale factor in order to not saturate the objective function.

Once car stop or slow down penalties are introduced, it is possible to categorize cars to consider also the pollution produced when the car does not proceed. Quantity and types of produced pollutants does not differ only for machine category (Petrol car, Gasoline car, Electric car) but also on the type of filter system mounted on the car.

Bibliography

[1] ”Near-real-time monitoring of global CO2 emissions reveals the effects of the COVID-19 pandemic”, written by Liu, Zhu and Ciais, Philippe and Deng, Zhu and Lei, Ruixue and Davis, Steven J and Feng, Sha and Zheng, Bo and Cui, Duo and Dou, Xinyu and Zhu, Biqing and others. Published in journal ”Nature communications”, in the year 2020, publisher Nature Publishing Group.

[2] ”The Ising Model”, written by Standford University.
https://stanford.edu/~jeffjar/statmech/intro4.html

[3] ”D-Wave Previews Next-Gen Platform; Debuts Pegasus Topology; Targets 5000 Qubits”, written by John Russell.
https://www.hpcwire.com/2019/02/27/d-wave-previews-next-gen-platform-debuts

[4] ”BreezoMeter documentation”.
https://docs.breezometer.com/api-documentation/air-quality-api/v2/#request-parameters

[5] ”Open Street Map”, general information about this tool.
https://en.wikipedia.org/wiki/OpenStreetMap

[6] ”OSMnx documentation”.
https://osmnx.readthedocs.io/en/stable/

[7] ”GitHub OSMnx open source repository”.
https://github.com/gboeing/osmnx

[8] ”About Us, D-wave system”.
https://www.dwavesys.com/company/about-d-wave/

[9] ”D-wave documentation”.
https://docs.dwavesys.com/docs/latest/c_gs_4.html

[10] ”Quantum Annealing in 2022: Practical Quantum Computing”, written by Cem Dilmegani.
https://research.aimultiple.com/quantum-annealing/

[11] ”Un record per la sovrapposizione quantistica”, an article from ”Le Scienze”, talking about quantum superposition.
https://www.lescienze.it/news/2019/10/02/news/molecola_record_sovrapposizione_quantistica-4568356/

[12] ”Introductive section of D-wave documentation”, talking about quantum annealing.
https://docs.dwavesys.com/docs/latest/c_gs_2.html

[13] ”Quantum superposition”.
https://it.wikipedia.org/wiki/Principio_di_sovrapposizione_(meccanica_quantistica)

[14] ”Quantum Tunneling Is Not Instantaneous, Physicists Show”, written by Anil Ananthaswamy, on July 22, 2020.

https://www.scientificamerican.com/article/quantum-tunneling-is-not-instantaneous-physicists-show/

[15] "Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models", written by Glover, Fred and Kochenberger, Gary and Du, Yu, on 2019, by Springer publisher.

[16] "Traffic flow optimization using a quantum annealer", written by Neukart, Florian and Compostella, Gabriele and Seidel, Christian and Von Dollen, David and Yarkoni, Sheir and Parney, Bob, on 2017 by Frontiers publisher.

[17] "COMPUTER QUANTISTICO, Il portale sul Quantum Computing", provide a general overview about quantum computers. http://computerquantistico.com/

[18] "Le sei sostanze inquinanti più pericolose su scala mondiale del 2015" https://greencross.ch/it/news-info-it/rapporti-ambientali/sei-sostanze-inquinanti-piu-pericolose-2015/

[19] "Conferenza di Rio de Janeiro 1992" https://www.ecoage.it/conferenza-rio-de-janeiro-1992.htm

[20] "Kyoto protocol" https://www.britannica.com/event/Kyoto-Protocol

[21] "The history of electrical car" https://www.energy.gov/articles/history-electric-car