Digitized Quantum Annealing for Logistic Network Design Problems

September 2021

Student:

Mikel Garcia de Andoin

Advisor:

Xi Chen

Co-advisor:

Mikel Sanz

Master in Quantum Science and Technology

eman ta zabal zazu

Universidad del País Vasco | Euskal Herriko Unibertsitatea

FACULTY OF SCIENCE AND TECHNOLOGY
UNIVERSITY OF THE BASQUE COUNTRY

Abstract

We propose a digitized version of the quantum annealing algorithm to solve logistic network design problems. Our algorithm employs improved heuristics for the selection of the parameters of the Hamiltonian encoding the problem. We also propose an evolution scheme which allows us to reach high fidelities with a circuit depth that grows linearly with the size of the problem I and the number of discretization steps n_T on an ideal quantum computer, $\mathsf{O}(I; n_T) = 2n_T I$. We present an alternative evolution scheme based on digital-quantum computation, which can be implemented in a realistic quantum computer with a limited connectivity among qubits. We introduce the tools required to implement this algorithm in both digital and digital-analog paradigms. The results shows that our algorithm succeed in obtaining a high fidelity for a predictable selection of n_T and total annealing times.

Contents

References **51**

1. Introduction

In the last years, quantum computing has attracted a lot of attention. The number of researchers working on this topic has drastically increased and new manuscripts are published every day showing promising results. To provide a perspective on current situation, we can compare it to classical computation, recalling us the time when CPUs started to be widely available. As the hardware got cheaper and more powerful, the development of new algorithms to tackle computationally hard problems blossomed. Nowadays, we enjoy the products of the work carried out decades ago. New problems that a few years ago seemed unfeasible to solve, now gather the attention of researchers. Many of these problems fall into the optimization problem category, one of which is the focus of this thesis.

The most relevant optimization problems are into the NP-hard complexity class. This means that a non-deterministic Turing machine could solve in polynomial time [1, pp. 463-465]. However, a classical algorithm deterministically solving an NP-hard problem takes a exponential time for it to complete. This limitation made researchers think on other algorithms to address these problems. Approximate algorithms reach a reasonably accurate (sometimes even the exact) solutions to these problems much faster. On this topic, metaheuristic algorithms have been shown useful for this task. Usually inspired by the nature, invented by researchers, or generated by a trained artificial intelligence, metaheuristic algorithms apply some optimization strategies which converges to the exact solution. The faster their convergence rate, the less time they need to yield an acceptable solution, which in many cases is su cient for a particular problem.

Here it is where quantum computing comes in as a possible improvement over classical computing. Due to the laws of quantum mechanics, quantum computing shows some useful properties that might help to speed up the convergence towards a solution for these hard problems. Entanglement and state superposition are the two main reasons why some quantum mechanical processes (i.e. quantum algorithms) take less time than their classical counterparts.

Although the field of quantum computing is relatively new, it has enjoyed a rapid growth. While the first ideas were proposed during the 80's [2, 3], the real kicko was in the 90's. On the side of quantum algorithms, Grover [4] and Shor [5] came up with algorithms that e ciently solve the problem of the unstructured search and prime factorization respectively. These results gathered a lot of attention, as they show quantum speedup for some useful problems. Since 2000, a large amount of articles and new ideas emerged. On the hardware side, the development of quantum platforms has progressed on a steady pace. Among the more developed ones we find trapped ions, photonics and superconducting circuits (SCC). This last platform is in the sight of most tech companies, with a number of them fostering the fabrication of a universal quantum computer, such as IBM, D-wave, Google, IQM... There are also some other proposals that seems promising on this topic even if they might be in earlier stages of development. These are quantum dots, non-linear optics, and topological qubits, just to mention some of them.

Within the framework of quantum computing, there are di erent paradigms. The closest one to the classical digital computing is digital quantum computing (DQC), which works with a set of ideal operations (or gates) which are applied in sequence. The similarity is evident, the problem is introduced in the quantum system as a set of controllable parameters and the solution is measured from the qubits once the gates are applied. Close to DQC, we have the digital-analog quantum computing (DAQC) paradigm, which takes advantage of the natural system's Hamiltonian to perform the operations in a more resilient manner. An alternative to this model is the adiabatic quantum computation (AQC), in which the problem is encoded into a ground state of a family of Hamiltonians. By means of an adiabatic process, a system starting from an easily preparable ground state evolves analogically from the initial to the problem Hamiltonian. Intimately related to AQC, we have the quantum annealing (QA) algorithm, which can be described as a metaheuristic for optimization problems. There is a debate on whether AQC is equivalent to QA, the consensus seems to be that the di erence between them is a relaxation on the condition of the adiabatic theorem. Indeed, QA can in principle be performed diabatically [6, 7].

This master thesis is placed on the crest of the wave of quantum computing development. The objective of this work is to provide a digitized version of a QA algorithm that solves a specific optimization problem. This problem is the logistic network design (LND) problem, which provides the optimal configuration for a supply chain connecting production centers with the final clients. In our globalized world, this logistic problem is of a great interest, since the production, transport, and distribution of goods is a major economic issue. Thus, the optimization of the supply chain can reduce unnecessary costs and greenhouse gasses emissions. The algorithm proposed in this thesis has the objective of overcoming some of the limitations shown by previous approaches. Since this task is so relevant and complicated, much further research is needed to address the many remaining open questions.

We start the thesis by introducing the fundamental concepts and tools required to follow the design of the algorithm (Ch. 2). We introduce some basic ideas about quantum computing and the two paradigms in which we will work. We briefly review the quantum annealing algorithm and the techniques required to implement it in a quantum computer in its digitized version. We also review the Trotter-Suzuki formula, which on top of providing us a technique to digitize the analog evolution, it gives the bounds for the error introduced in this process.

Then, we present our specific LND problem and some of the previous approaches for solving it (Ch. 3). These previous algorithms include, along other classical algorithms, a hybrid classical-quantum algorithm which is the inspiration for our algorithm. Chapter 4 is the core of the thesis, since it is where we introduce our algorithm in detail. The algorithm developed here is, to our knowledge, the first algorithm to perform the digitized version of a quantum annealing process using a digital-analog scheme on a SCC with limited coupling topology. We also improve the penalty factors previously used, increasing the fidelity of the algorithm. We apply the tools mentioned in chapter 2 to implement the algorithm in a quantum circuit with limited topology, making it feasible to be run with realistic devices. By using an small problem, we show how this algorithm successfully solves a LND problem by simulating an ideal system (Ch. 5).

Finally, we comment the results obtained and the research left to do. The most important pending task is to check the validity of the algorithm in a noisy quantum computer. One of the possible improvements might come from applying shortcuts to adiabaticity (STA) strategies.

2. Fundamental concepts

In this chapter we review the theoretical aspects and tools on which this master thesis is based. In this chapter, we first introduce some basic concepts about the DQC paradigm, along with a brief introduction about the physical realization in superconducting circuits (SCC). Then, we introduce both QA and DAQC, on which our algorithm is based. As the algorithm is a digitization of a QA algorithm on the DAQC framework, we describe the techniques required to transform it. Finally, we explain the Trotter-Suzuki formula, which allows us to correctly digitize the annealing process and characterize the errors.

2.1 Digital quantum computing (DQC)

Digital quantum computing is a computing paradigm that uses qubits to encode the information and quantum gates to perform the operations established by an algorithm to obtain the solution to a problem. This resembles classical computing in the sense that the information is coded into bits and the operations are implemented by means of Boolean logic gates.

A qubit is a two dimensional quantum system whose quantum state can be represented as a linear combination of the states of its Hilbert space basis. In the canonical basis, we can use the ground $|0\rangle$ and the excited $|1\rangle$ states to describe any qubit, $|\ \rangle = |0\rangle + |1\rangle$, where $|\ \rangle$ is a normalized state, $\langle\ |\ \rangle = 1$, and $;\ \in \mathbb{C}$. For systems comprising n qubits, this representation is extended by using the basis of the Hilbert space of dimension 2^n. This way, a state can be represented by a normalized vector of 2^n complex coe cients. However, this representation is only valid for pure states. The general approach to describe any quantum state, pure or mixed, is to use the density matrix representation. For a system with n qubits in a pure state $|\ \rangle$, its density matrix is a $2^n \times 2^n$ matrix constructed as $= |\ \rangle\langle\ |$. This representation is used when we have to take into account the e ects of non unitary phenomena, i.e. the di erent sources of noise. In this thesis, we will restrict ourselves to pure states and their ket representations, unless otherwise stated.

There is a geometric representation, called Bloch sphere, which is useful to visualize the state of a single qubit and the e ect of di erent gates on it. For a qubit in a pure state, the normalization condition imposes a constraint on the values of the coe cients of the vector representation, $|\ |^2 + |\ |^2 = 1$. As and are complex numbers, this describes the coordinates of the surface of an unit sphere in a three dimension space. The coordinates that intersect the axes corresponds to the eigenstates of the Pauli operators. An example of the Bloch sphere is shown in Fig. 1.

Quantum gates are operators applied to the qubits and whose sequence implements a quantum algorithm. A gate $\hat{A}$ can be described by means of a unitary matrix, i.e. a matrix A that fulfils $AA^\dagger = A^\dagger A = \mathbb{1}$. For simplicity, from now on the operators and their matrix representation will be written both the same, unless it is necessary to make a clear distinction. One important characteristic of quantum gates is the reversibility, i.e. the e ect of a gate can be reversed by applying its conjugate transpose. The e ect of applying a quantum gate is obtained by the time evolution of the quantum state under a certain Hamiltonian. This can be written as

$$|\ '\rangle = A|\ \rangle = e^{\ iHt=\sim}|\ \rangle;\quad {}' = A\ A^\dagger = e^{\ iHt=\sim}\ e^{iHt=\sim}; \tag{1}$$

We can now easily prove the reversibility of a quantum gate by just writing $A^\dagger|\ '\rangle = A^\dagger A|\ \rangle = |\ \rangle$.

In general, a quantum algorithm can be understood as a global unitary operator that is applied to all qubits in the system, which transforms the initial state into the state coding the solution. However, in order to apply the global unitary operator in a realistic physical system, we need to use a sequence of simpler gates. As in classical computation, in quantum computation there exist universal sets of gates from which we can e ciently construct any other gate (see Slovay-Kitaev theorem [8]). The most simple

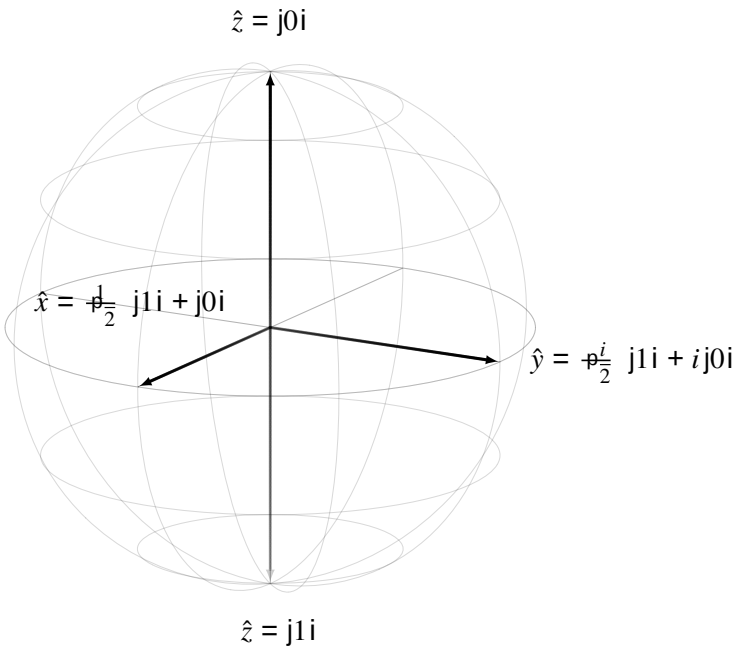


Fig. 1: Bloch sphere visualization of a qubit. The pure states used in quantum computing fall into the surface of the unit sphere. The states of the basis of the computational basis are represented in the poles of the sphere, while every other state is a combination of both.

universal set of gates is composed with single-qubit gates (SQG) and one two-qubit gate [9]. Although there are infinitely many gates, some are specially useful and widely used. Here we introduce the ones required to follow this thesis:

Pauli gates: These SQGs apply the Pauli operators to a qubit. In the Bloch sphere representation, the e ect is a rotation around the corresponding axis. They are usually referenced to withdenoted by capital letters X, Y, Z or with either $_i$ or ^{i}, with $i = \{x, y, z\}$. The ^{x} gate is also called NOT, because it performs the same operation as the classical NOT gate when acting on the computationalcanonical basis $\{|0\rangle, |1\rangle\}$. The matrix representation in the canonical basis of the Pauli gates is

$$X = {}^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; \; Y = {}^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}; \; Z = {}^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2}$$

From this point on, we will use the canonical basis for the representation of the quantum gates.

Rotation gates: While the Pauli gates only allow for rotations with a fixed angle, the rotation gates are the generalization for an arbitrary angle. As any operator that acts on one qubit can be written with two independent variables, any single qubit gate can be written with two rotation gates. This is easy to justify by thinking on Euler angles for any rotation on 3D. The matrix representation for these gates is

$$R_x(\theta) = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}; \; R_y(\theta) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}; \; R_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}. \tag{3}$$

The rotation gates corresponds exactly to the evolution of the qubit with a Hamiltonian with local Pauli terms, this is $\exp(-i\theta\, {}^i) = R_i(2\theta)$. This equivalence will be later useful to simulate evolution operators using the digital quantum computing framework.

Hadamard gate: It is one of the most used gates, as it creates an equal superposition of the computational basis states. It is the composition of a 90º rotation around the y axis, followed by a

180º rotation around the x axis. This gate is labeled with H, and its matrix representation is

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{4}$$

Two-qubit rotation gates: The generalization of rotation gates to two-qubit spaces around the set axis in the two-qubit Bloch sphere. These gates are constructed as the evolution of a Hamiltonian which has a coupling term consisting on the tensor product of two Pauli matrices. We can construct this continuous set of gates using a universal set of gates. For example, we can simulate the evolution of a two-qubit term $_z\ _z$ with constant coupling by employing two two-qubit gates and one SQG [10], shown in Fig. 2.

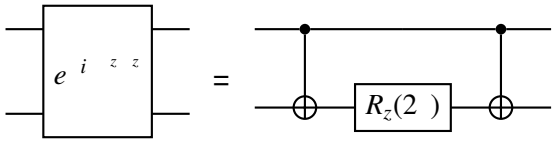


Fig. 2: Circuit that performs the simulation of the evolution of a $^z_1\ ^z_2$ Hamiltonian using two CNOT gates and a SQG. The representation used corresponds to the quantum circuit model.

Swap and iSwap gates: These gates interchange the states of two qubits. The di erence between both is that the iSwap gate introduces a phase to the state, while the Swap gate just interchanges the state of the qubits. The matrix representations are

$$\text{Swap} = \begin{pmatrix} 1 & & & \\ & 0 & 1 & \\ & 1 & 0 & \\ & & & 1 \end{pmatrix}; \quad \text{iSwap} = \exp\left[\frac{i}{4}(_x\ _x + _y\ _y)\right] = \begin{pmatrix} 1 & & & \\ & 0 & i & \\ & i & 0 & \\ & & & 1 \end{pmatrix}; \tag{5}$$

where, from now on, the missing matrix elements are zeros. A Swap (iSwap) gate acting between the qubits i and j of the system will be referred to as Swap$_{ij}$ (iSwap$_{ij}$).

The advantage of the iSwap gate over the Swap gate is that it can be implemented by using a Hamiltonian with one $_i\ _i$ coupling, in particular with $_z\ _z$. We can do this on on various ways. We start by splitting the exponential in two terms, $\exp[(i\ =4)(_x\ _x + _y\ _y)] = \exp(i\ _x\ _x=4)$ $\exp(i\ _y\ _y=4)$. Then, simple way to implement each term this by is rotating the qubits, performing the evolution, and then rotating the qubits back. For example, to simulate the evolution of the $_y\ _y$ term, we can use the circuit on Fig. 3. The circuit for the $_x\ _x$ term can be obtained in a similar manner, in this case by employing rotations on the y axis.

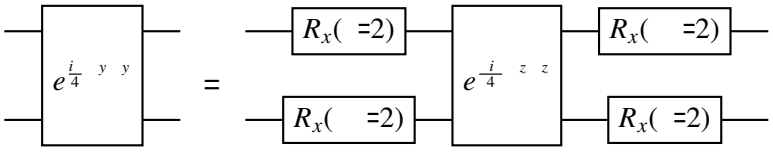


Fig. 3: Circuit that performs the simulation of the evolution of a $_y\ _y$ Hamiltonian using SQGs and a $_z\ _z$ source. The iSwap gate can be implemented by concatenating this and the circuit simulating the evolution of $_x\ _x$.

We have already used the quantum circuit model to describe some quantum gates in Figs. 2,3. This model of computation describes the gates and the sequence at which they are applied to the qubits. For this, we assign each qubit a horizontal line. The gates are depicted as rectangles that span over the qubits they act on, and the time at which they are employed goes from left to right. Although there are more rules to describe any algorithm, these are su cient to follow this thesis.

The order at which the gates are applied is important for the outcome of the algorithm to be correct. If two gates commute, then we can apply them in any order. This is true for example for any two gates that are applied on two qubit subsets that do not overlap. This case in particularly useful, since gates that act on di erent subspaces can be simultaneously applied. But for non-commuting gates $[A; B] \neq 0$ the order must be preserved, as $AB \neq BA$.

In order to run a quantum algorithm there are some steps to follow. First, the system must be initialized in an easily preparable state. This is usually the ground state of the system, $|0\rangle^{n}$. Then, after applying the gates, the information about the outcome of the process must be obtained, i.e. the qubits must be measured. However, the full information about the state of the qubit is practically inaccessible. Experimentally, the measurement of a qubit, is usually made by projecting the qubit state onto the z axis of the Bloch sphere, with theleading to only two possible outcomes, namely $|0\rangle$ and $|1\rangle$. If we wanted to extract more information about the final state, we would need to repeat the process in other bases so that we obtain statistical information about the distribution of the possible observables. In other words, this means measuring the system in the projection of the states of the canonical basis $\{|0\ldots0\rangle\langle 0\ldots0|, |0\ldots01\rangle\langle 0\ldots01|, \ldots, |1\ldots1\rangle\langle 1\ldots1|\}$, which is essentially the classical information of the system. This process is called quantum tomography.

2.1.1 Physical realizations of DQC

In order to implement a quantum algorithm we need a quantum hardware, this is, a quantum computer. Although there are many di erent ways to realize a qubit, in this work we will focus just on superconducting circuits (SCC). SCC take advantage of the quantum properties of superconductivity to create anharmonic quantum oscillators. Then, a superconducting qubit can be realized by taking two of the states of these oscillators. A crucial advantage of SCC over other platforms is that the circuit parameters are tunable up to the limits of the fabrication process, which resembles the ones used in the fabrication of usual semiconductor chips. This allows us to precisely engineer a system to work in the desired frequencies, that usually spans from the radio to the microwave spectrum [11, 12].

Qubits in SCC processors are usually constructed starting from a LC circuit with a harmonic oscillator Hamiltonian. If we work with the energy states of a harmonic Hamiltonian, we cannot address correctly each level separately, as we cannot predict the system's final state after addressing the system with a pulse with the gap frequency. In order to control the system level by level, we need to separate them with di erent energy gaps. In order to achieve this, we require an anharmonicity obtained by introducing a non-linear element called the Josephson junction (JJ). The geometry of the JJ is similar to a classical planar capacitance, with two opposing superconducting electrodes which sandwich an insulator. This gives the JJ a intrinsic capacitance C_J equivalent to the ones of the classical elements. The non-linearity of the JJ shows on the equation for the supercurrent, on what is called the DC Josephson e ect, $I = I_C \sin$, where is the phase di erence of the voltage across the junction and I_C is the critical supercurrent that depends on the material. The circuit equations yields a non-linear voltage-current relation, given by the non linear inductance $L = \hbar/(2e\, I_C \cos$). Thus, a JJ can substitute a classic element in a resonator to create the anharmonic oscillator. A usual parameter to characterize the JJ is the ratio E_J/E_C, where $E_J = \hbar I_C/(2e)$ is the Josephson energy and $E_C = e^2/(2C)$ the electrostatic Coulomb energy [13].

There are mainly three types of qubits in SCC: phase qubits, flux qubits and charge qubits. For simplicity and convenience, in this thesis we will only work with the former, in particular with the transmon qubit. The principle behind this qubit is the appearance of extra cooper pairs in the island

formed by one of the electrodes, the capacitor and the JJ. To deduce the Hamiltonian, we can start by writing the circuit equations in a convenient choice of conjugate variables, the flux and the charge Q, instead of the usual voltage-current pair used in electronics. The Hamiltonian for a linear circuit of a capacitor and an inductance in parallel can be written with the usual procedure,

$$H = \frac{1}{2}CV^2 + \frac{1}{2}LI^2 = \frac{Q^2}{2C} + \frac{^2}{2L}; \tag{6}$$

where the flux and the charge are obtained from the time integral of the voltage and the current respectively,

$$(t) = \int_0^t V(t')dt'; \; Q(t) = \int_0^t I(t')dt': \tag{7}$$

Now, we make a further change of variables, introducing the reduced flux $= 2 = _0$ where $_0 = h/(2e)$ is the superconducting magnetic flux quantum, and the reduced charge $n = Q/2e$, with e the electron charge. If, instead of a linear inductor, we use the nonlinear JJ as shown in Fig. 4, the resulting equation for the circuit is

$$H = 4E_C n^2 \quad E_J \cos( ); \tag{8}$$

where the electrostatic Coulomb energy now depends on the capacitance in series of the capacitor C_g and the JJ C_J, $E_C = e^2/(2(C_g + C_J))$.

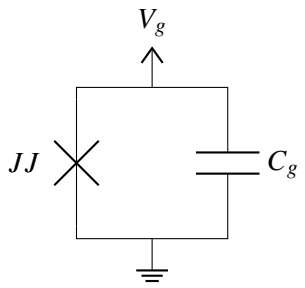


Fig. 4: Diagram for the superconducting circuit of the transmon, which is a particular type of charge qubit. This circuit has the same form as a resonator where the linear inductance is substituted by the non-linear JJ, usually depicted with a cross, obtaining an anharmonic Hamiltonian.

By designing the fabrication characteristics, we can modify the relation between the energies E_J and E_C. The optimal working range for the charge qubits is $E_J \quad E_C$, which reduces the sensitivity to charge noise and it is easier to produce with current technology. Then, since we are working with small values of , we can expand the second term of the Hamiltonian as a power series around $= 0$,

$$E_j \cos( ) \quad E_j \; 1 \quad \frac{^2}{2} + \frac{^4}{24} + O(^6) \quad \frac{E_J}{2} \; ^2 \quad \frac{E_J}{24} \; ^4; \tag{9}$$

where we have taken out the constant energy term. Working up to the forth order, we can now write the complete Hamiltonian as an anharmonic oscillator. For this, we can threat the flux as the generalized position coordinate and the charge as the generalized momentum, this is $[\;;Q] = \sim[\;;n] = i\sim$. As this is the case, we can rewrite the operators in terms of the creation and annihilation operators, $n \; / \; i(a \quad a^\dagger)$ and $\; / \; (a + a^\dagger)$.

To obtain a compact expression for the two level system, we define two new parameters. The qubit frequency $_q = (\sqrt{8E_J E_C} \quad E_C)/\sim$ is the frequency required to excite the system if it were just an harmonic oscillator. The anharmonicity takes into account the frequency difference to excite the system from the ground to the first excited and from the first to the second excited, $= _q(e_1 \; e_2) \quad _q(g \; e_1) = E_C$. Ideally we would want this anharmonicity to be sufficiently large to model the system as a

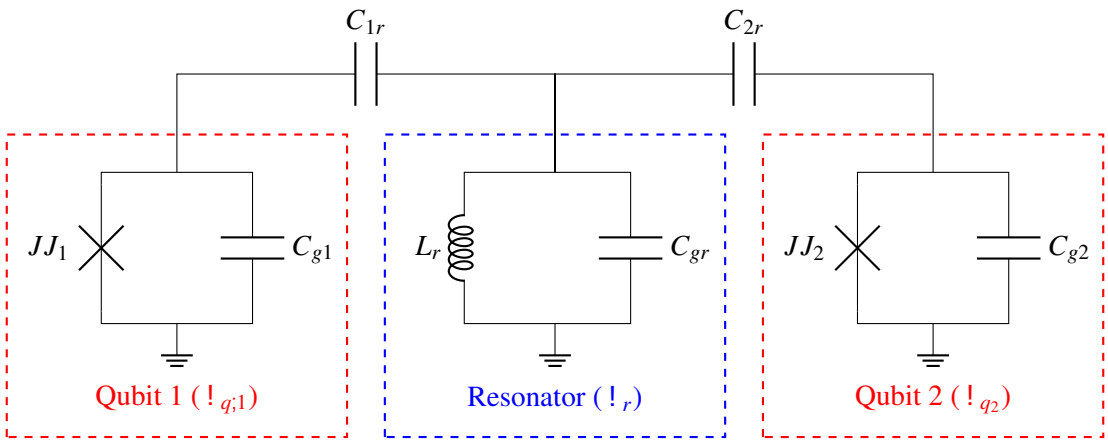


Fig. 5: Schematic for the circuit that implements a capacitive coupling of two transmon qubits with a resonator. The transition frequency of each qubit is ! $_{q;i}$ and the resonator frequency is ! $_r$. At the dispersive limit, the resonator is e ectively isolated from the qubits, leaving just a direct interaction between the qubits.

two level system. For this condition to hold, the circuit is usually set to work in the range of E_J=E_C 10. Then, we can write the qubit Hamiltonian from Eq. 4 with the approximation from 9 as

$$H \quad !_q a^y a + \frac{}{2} a^y a^y a a \quad !_q a^y a = \frac{!_q}{2} \; ^z + \frac{!_q}{2} \mathbb{1}: \tag{10}$$

What we have found after this discussion is that the local term of the transmon qubit is a ^{z} term.

Now we will briefly introduce the coupling terms between qubits. In order to keep the explanation simple and since is the coupling used in the algorithm, we will just explain the transverse coupling of two charge qubits. For the case of the transverse coupling, it emerges in the Hamiltonian as a $^z_1 \; ^z_2$ term. One way to obtain these terms is to physically couple the qubits with an intermediate resonator. To address each qubit separately we need to add an element that physically separates both qubits. This can be done with mutual inductance, or more conveniently, with a capacitor between each element of the circuit. An scheme of this design is shown in Fig. 5.

In general, the Hamiltonian for a system with two interacting elements can be written as the sum of each term

$$H = H_1 + H_2 + H_{int}; \tag{11}$$

where H_i for $i = 1; 2$ are the Hamiltonian of the qubits and H_{int} is the interaction Hamiltonian. As we are working with two transmon qubits, each of the terms H_1 and H_2 will take the form of Eq. 10.

Here, the interacting term corresponds to the resonator. If we start from the Hamiltonian of the harmonic oscillator in Eq. 6 and apply the procedure to quantize it, we will end up with the well-known quantum harmonic oscillator Hamiltonian. But now, as each qubit is connected to the resonator, an excitation from a qubit can jump to it and vice versa. If we take for granted that the qubits are perfect two level systems, the Hamiltonian of the system can be written as

$$H = \sum_{i=1;2} !_{q;i} a_i^y a_i + g_{ir} \; a_i^y a_r + a_i a_r^y \quad + !_r a_r^y a_r; \tag{12}$$

where ! $_r$ is the frequency of the resonator and g_{ir} the coupling strength between each qubit and the resonator. If we work in the dispersive limit, i.e. g_{ir} j ! $_{q;i}$! $_r$j, the resonator can be considered as an isolated system. Then,, only terms corresponding to the two qubits remain,

$$H \quad \frac{!_{q;1}}{2} \; ^z_1 + \frac{!_{q;2}}{2} \; ^z_2 + g \; ^z_1 \; ^z_2; \tag{13}$$

where g is the coupling strength between the qubits.

This simple circuit can be repeated for more qubits, in a circuit in which a qubit is coupled to its neighbour. However, this might not be useful for a real implementation. A similar coupling can be obtained by substituting the classical resonator with a SQUID, which is the parallel of two JJ [14]. The main advantage of this design over the previous one is the control gained on the interaction, allowing for turning it on and o , and producing a longitudinal coupling in an arbitrary axis.

A constraint that is di cult to quantify is related to the topology of the circuit. Ideally, a quantum system would have its qubits all-to-all (ATA) connected by pairs. By having any two qubits connected, all two-qubit gates could be directly implemented. However, this kind of circuits are not doable with current fabrication techniques. The topology of the couplings of the qubits is usually limited no more than 4 couplings per qubit, as it is the case for IBM's computers [15] and D-wave's annealers [16].

Let us briefly describe how the gates are applied. This is a complex topic which is not in the core of this thesis. Nevertheless, it is interesting to grasp some fundamental ideas about how to apply them and the limitations we have to overcome. Briefly speaking, quantum gates are applied by sending electromagnetic pulses to the qubits. The signals sent to the qubits are composed by a carrier frequency of the frequency of the qubit, modulated by a lower frequency signal. The modulating signal generates a waveform of the length and shape required to apply the desired gate. For instance, an X gate can be implemented using the typical pulse. This is done by modulating the carrier frequency signal with a square pulse. But while an ideal square pulse will apply the operation we were aiming to, in reality the pulses we can send to the qubits are smoothed, so the results of applying a gate can di er from what it is expected.

Furthermore, the physical realizations of the qubits and the gates have some other limitations. On the one hand, we have fabrication defects. As in any circuit constructed in the nanometric scale, a small defect can drastically change its behaviour [17]. On the other hand, the control techniques applied to the system are far from ideal and are subjected to di erent noise sources. We can distinguish four main noise sources for a quantum computer [18]:

Longitudinal relaxation: This is the e ect on a qubit of randomly flipping between the ground and the excited state. The rate at which this flip occurs is measured with the decay time $T_1 = (\ _{1\#} + \ _{1"})^{\ 1}$, where $\ _{1\#}$ is the rate at which the transition from the excited to the ground state occurs and $\ _{1"}$ the rate for the opposite transition. Generally, $\ _{1"}$ is exponentially suppressed by the temperature, which is usually in the order of tens of milikelvin. In the currently available computers from IBM the relaxation time is from 1 to 100 microseconds.

Dephasing: This refers to the depolarization in the xy-plane of the Bloch sphere. It is measured by the characteristic time $T_{\cdot} = \ _{\cdot}^{\ 1}$, which is the time for the state to decohere from a pure state on this plane to a classical mixture.

Transverse relaxation: It describes the decoherence of a superposition state, which is the joint e ect of the longitudinal relaxation and the dephasing. It is characterized by the transverse relaxation time $T_2 = \ _2^{\ 1} = (\ _1/2 + \ _{\cdot})^{\ 1}$. On IBM computers this time is in the same order of magnitude as the relaxation time, $T_2 \quad 1 \quad 100 \quad s$.

Gate errors: This refer to the how close to ideal the e ect of a gate is on the system. Among other reasons, this can be attributed to the aforementioned realistic control of the qubits. One of the metric employed to estimate the rate at which gate errors occur is the fidelity. This measures how close is the actual evolution of the system and the ideal gate that was expected to be applied [19]. Although these number could vary from one computer to another, we can take a particular computer to take a grasp of the state-of-art. For instance the *ibmq_manila* computer has SQG errors on the order of magnitude of 0.01%, while the error for two-qubit gates grows up to 0.1%. Gate

errors are widely referred to by the fidelity, i.e. the probability of applying the gate successfully. In this case, the error rates just commented yields a fidelity of 99.99% and 99.9% respectively.

All these errors can, among some others, produce qubit-flip errors. A qubit state can make a change of direction on the Bloch sphere representation in any axis. To correct these errors, there are di erent error correction codes. They work in a similar way as the classical error correcting codes. The main idea behind them is to duplicate the information along more than one qubit so that errors can be detected and/or corrected [20]. There are also control techniques that help mitigating some errors, like amplification techniques [21] or dynamical decoupling [22]. However, even though we use these techniques, running an algorithm in a real quantum computer is still subjected to errors. Thus, it is relevant to check how an algorithm behaves when the actual run is not completely ideal.

2.2 Quantum annealing (QA)

Before stepping into the quantum annealing (QA) algorithm, let us briefly introduce the fundamentals behind the classical annealing algorithm. Simulated annealing (SA) is a global optimization algorithm inspired in the annealing process in metallurgy [23]. SA takes a problem in terms of a real function that works as a energy potential and an initial temperature T. The system starts in a random initial state. Then, in each iteration, the system has a probability of jumping to another state. This probability grows both with the temperature and with the di erence between the new and the current state energies. This makes the system more likely to jump to states with less energy, and thus makes it more likely to arrive at the global minimum. The selection of the candidate for the jump is made by proximity to the current state, making more probable to jump to neighbouring states. After each iteration, the system's temperature gets reduced until it reaches zero, the system can no longer jump to other state and the algorithm stops. A provable characteristic of SA is that if the cooling process is made in infinitely many steps, the algorithm is guaranteed to reach the optimal solution.

SA shows some relevant problems. The more obvious one is that the number of steps of the annealing must be finite, thus it is no longer guaranteed to reach the optimal solution of the problem. More important is the algorithm capacity to prevent the convergence towards a local minimum. This can happen when a state is trapped in a potential well, surrounded by energy barriers. The probability for the state to cross a energy barrier is drastically lowered if the barrier is wide or high enough.

Quantum Annealing (QA) o ers a feasible solution to the last problem by taking advantage of the quantum property of tunneling and the adiabatic quantum computing (AQC) scheme. QA takes a slightly di erent approach to the classical cooling process. The algorithm starts in the ground state of an initial Hamiltonian (H_0). The optimization problem is encoded into a target Hamiltonian (H_1), where its ground state is the global optimum. In this case the, annealing process is made by transitioning the system from H_0 to H_1. A possible manner to perform this transition is by implementing the following time dependent Hamiltonian

$$H(t) = (1 \quad (t))H_0 + \quad (t)H_1; \tag{14}$$

where $(0) = 0$ and $(T) = 1$, with T the total time for the process. Therefore, the time dependent Hamiltonian fulfils $H(0) = H_0$ and $H(T) = H_1$. The simplest function that we can consider is a linear function $(t) = t=T$.

The key point of QA is that, starting from the initial ground state, if this transition is adiabatically performed, the state of the system will remain the ground state during the evolution. In particular, the system will also remain at the ground state at the end of the process. Thus, measuring the final quantum state yields the solution to the problem.

Theorem 1 (Adiabatic theorem [24, 25])**.** *A time-dependent quantum system remains in its instantaneous eigenstate, in particular its ground state, at all times if it is not degenerated at any point of the evolution and the evolution under the Hamiltonian is performed infinitely slow, i.e. adiabatically.*

Experimentally, it is impossible to perform an evolution infinitely slow. Fortunately, the adiabatic theorem give us a technique for estimating a bound for the evolution time required to have a high probability of remaining the ground state. Indeed, the minimum evolution time T must satisfy the condition [26]:

$$T \quad \frac{\mathrm{j}\dot{H}(t)\mathrm{j}_{max}}{E^2_{min}}; \tag{15}$$

where

$$\mathrm{j}\dot{H}(t)\mathrm{j}_{max} = \max_{0 \; t \; T} \; \mathrm{h} \quad _0(t)\mathrm{j} \frac{dH(t)}{d(t=T)} \mathrm{j} \quad _1(t)\mathrm{i} \; ; \tag{16}$$

and

$$E^2_{min} = \min_{0 \; t \; T} (E_1(t) \quad E_0(t))^2: \tag{17}$$

Here, $_0(t)$ and $_1(t)$ are the instantaneous ground and first excited states respectively, and $E_0(t)$ and $E_1(t)$ their energies.

Now, the only part left is the codification of the optimization problem into a Hamiltonian. First, we have to take into account with which kind of Hamiltonian we can work. Ideally, we could work with any kind of Hamiltonian. However, we are limited by the constraints of the physical devices. For instance, D-Wave's quantum annealers implement a Ising Hamiltonian of the form

$$H(t) = \frac{A(t=T)}{2} \sum_i \; ^x_i + \frac{B(t=T)}{2} \left(\sum_i h_i \; ^z_i + \sum_{i>j} J_{ij} \; ^z_i \; ^z_j \right): \tag{18}$$

The problem must then be codified with the local term parameters h_i and the couplings J_{ij}. The particular codification depends on the characteristics of the problem, but we present here some general guidelines. The input parameters for a problem can be given as a binary array of n bits, $q = [q_i]$ with $q_i = \mathsf{f}0; 1\mathsf{g}$ and $i = 1; : : : ; n$. Suppose the problem can be stated as finding the solution for an equation with all variables on one side, $F(q) = E$. Indeed, this corresponds to a set of Boolean closures. The function $F(q)$ can be in principle arbitrary. But, for now, let us consider that it can be decomposed into local terms (q_i) and interacting terms ($q_i q_j$). Then, the problem can be transformed into a minimization problem of the square of the di erence between both sides of the equation, $G(q) = (F(q) \quad E)^2$. Now, the only step left is to write this as a Hamiltonian. For this, we will make the following transformation: $q_i \; ! \; (\mathbb{1} \quad \; ^z_i)=2$. For terms with two bits, this is $q_i q_j \; ! \; (\mathbb{1} \quad \; ^z_i \quad \; ^z_j + \; ^z_i \; ^z_j)=4$. After applying this transformation, the expression has the same form as the second term of the Ising Hamiltonian, only with a constant energy di erence which can be always discarded by choosing a convenient energy gauge.

However, the constrained problems introduce some conditions on the allowed solutions. This constraints can be given as equalities or inequalities. For the first case, the condition can be expressed as an Ising Hamiltonian following the aforementioned procedure. For the second case, the inequality must first be converted into an equality. There are several ways of doing this. But, in this thesis, we will add slack variables. If an inequality can be expressed as $G(q) \quad g$, then there exists a variable $s \quad 0$ such that $G(q) \quad g + s = 0$. This converts the constraint problem with input q to a unconstrained problem with input $q \quad s$. It is guaranteed that the new problem with the slack variable solves the original problem when extracting the original input parameters [27, sec 4.1.3]. The slack variable s can be expressed with m binary variables. In particular, for problems where $0 \quad G(q) < g$ and $G(q); g \; 2 \; \mathbb{Z}$, s can be expressed in its binary representation with $m = \log_2(g)$ bits. This is precisely the case for finding the configurations of q that fulfil the inequality constraint. Then again, we can transform this term to $(G(q) \quad g + s)^2$ and express it as an Ising Hamiltonian, now with $m + n$ qubits. When added to the original problem, these new Hamiltonians are multiplied by a penalty, so that any state which does not fulfil the constraint is penalized with some extra energy. A naive choice of this penalty would be to set it to infinity, so that the system can never reach solutions that are not allowed. Indeed, there are more convenient choices for the penalty, as we will explore in this thesis for the case of the LND problem.

2.3 Digital-analog quantum computation (DAQC)

In contrast to DQC where the evolution is performed with a set of discrete transformations, i.e. the quantum gates, DAQC uses a slightly di erent approach. DAQC can be defined as the following: a quantum computation paradigm in which the state is evolved through a sequence of SQGs and analog blocks. An analog block is an operator that acts on the system as the evolution of a Hamiltonian, which is indeed the Hamiltonian that arises naturally in the quantum system. Then, by turning on and o the couplings, we can leave the system evolve under the interacting Hamiltonian.

This paradigm is specially useful in state-of-art of quantum computers. As noted before, the noise in quantum computers is a problem that needs to be addressed. Unwanted couplings between qubits are one of the sources of noise and errors. Instead of trying to reduce their e ect, these couplings might be useful to implement analog blocks. For instance, as we will see later in this thesis, the two qubit interaction that naturally arises in SCC commented in section 2.1.1 can be used to implement the Ising Hamiltonian instead of using only quantum gates.

2.3.1 Simulation of an ATA Hamiltonian using a NN Hamiltonian as a source

As previously advanced, DAQC scheme can be used, among other aims, to enhance the connectivity of a quantum computer. In principle, the capacity of a system to simulate the evolution through a Hamiltonian is limited by the couplings between the qubits. If we assume that the terms to which we have access are local and two-qubit interactions, then the topology of the circuit would impose that some of the coupling terms in the Ising Hamiltonian must be zero. This limitation comes from the fabrication process and the di culty to reach full connectivity.

However, the problems we want to address might not be straightforwardly codified in a given quantum processor architecture. For example, if a problem is codified into an ATA Hamiltonian, then a natural solution would require to have couplings between every two qubits. To optimize the available resources and to try to solve a larger set of problems, it is proposed to simulate an ATA Hamiltonian using a system with a nearest neighbour (NN) couplings configuration [28]. In this thesis, we will only focus in the simulation of ATA Ising Hamiltonians using a system with a NN configuration with ZZ couplings.

The first step is to decompose the ATA Hamiltonian into the sum of NN Hamiltonians. In graph theory, this problem is known as the partition of a complete graph into a set of Hamiltonian cycles, which are closed loops which visit once every graph node. This problem was solved in 1890 by Walecki [29]. The proposed solution consists on iteratively rotating a Hamiltonian cycle until the graph was completed. The solution for our particular problem is slightly di erent, as our source is a Hamiltonian path, which is a Hamiltonian cycle but with a missing edge. To make the solution more clear, we can label our qubits with numbers. Then, the graph of the source Hamiltonian can be represented by employing a vector $[1; 2; \dots; n]$, where n is the total number of qubits and each of the qubits is coupled with their neighbour, with a total of $n - 1$ couplings. Then, the couplings that we initially have are between qubits $1 \$ 2$, $2 \$ 3$, and so on. The decomposition consists of a series of qubit permutations $P^k(n)$, with $k = 1; \dots; \lfloor n/2 \rfloor$, which are constructed as

$$P^k(n) = \begin{cases} 1 + k \quad 1 + \frac{j}{2} \mod n; & \text{for } j \text{ even} \\ 1 + k \quad 1 \quad \frac{j-1}{2} \mod n; & \text{for } j \text{ odd} \end{cases}. \tag{19}$$

If n is even, the solution above is correct. However, if n is odd, some of the edges appear twice. In order to solve this problem, we propose to eliminate one every two edges starting from the second of either the first or the last term.

Now we need a method to perform these permutations to transition from one NN Hamiltonian to another. To attain this, we can use any of the two Swap gates presented in section 2.1. But we can also use the more realistic option of using iSwap gates. A permutation of two neighbouring qubits can be

achieved with a single iSwap gate. To permute two qubits separated by another qubit, we need to apply three iSwap gates: $\mathrm{iSwap}_{1;2}$, $\mathrm{iSwap}_{2;3}$ and $\mathrm{iSwap}_{1;2}$. This is shown in Fig. 6 for up to a distance of three qubits. In general, for a permutation of two qubits separated by m qubits we need to apply $2m + 1$ iSwap gates.

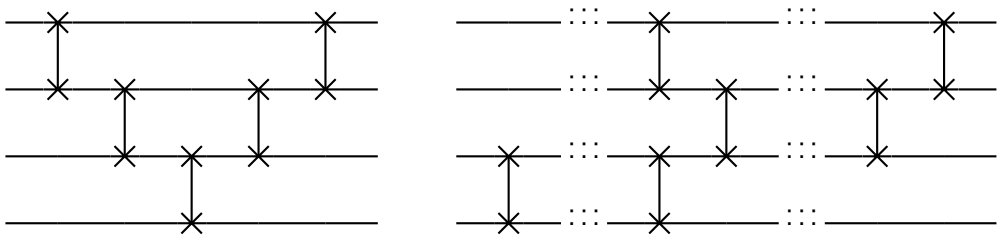


Fig. 6: Different permutation schemes. On the left, a permutation of two qubits separated by one qubit. On the right, the permutations from Eq. 20 for a 4-qubit system.

We can optimize the number of gates and steps needed by looking at the qubit distribution at each step of the process. Then, we can efficiently merge all the gates into blocks that apply different Swap gates at the same time. The resulting blocks depends on the parity of the number of qubits the system has,

$$
\begin{aligned}
F(0;\text{even } n) &= \prod_{j=n/2}^{2} \Big(\prod_{i=2j-2;\, j,2;\, i \text{ even}}^{n-1} \mathrm{iSwap}_{i;i+1} \prod_{i=2j-1;\, i \text{ odd}}^{n-1} \mathrm{iSwap}_{i;i+1} \Big); \\
F(0;\text{odd } n) &= \prod_{j=\lceil n/2 \rceil}^{2} \Big(\prod_{i=2j-3;\, i \text{ odd}}^{n-1} \mathrm{iSwap}_{i;i+1} \prod_{i=2j-2;\, i \text{ even}}^{n-2} \mathrm{iSwap}_{i;i+1} \Big); \\
F(k;n) &= \prod_{i=2k+2;\, i \text{ even}}^{L-1} \mathrm{iSwap}^{\dagger}_{i;i+1} \prod_{i=2;\, i \text{ even}}^{2k-1} \mathrm{iSwap}_{i;i+1} \prod_{i=2k+1;\, i \text{ odd}}^{L-1} \mathrm{iSwap}^{\dagger}_{i;i+1} \prod_{i=1;\, i \text{ odd}}^{2k} \mathrm{iSwap}_{i;i+1}; \\
F\left(\frac{n}{2};n\right) &= \prod_{j=1}^{\lceil n/2 \rceil - 2} \Big(\prod_{i=1;\, i \text{ odd}}^{n-2j} \mathrm{iSwap}^{\dagger}_{i;i+1} \prod_{i=2;\, i \text{ even}}^{n+1-2j} \mathrm{iSwap}^{\dagger}_{i;i+1} \Big);
\end{aligned}
\tag{20}
$$

where $k = 1;\ldots;\lceil n/2 \rceil - 1$. In order to implement this gates in DAQC, the Swap gates must be decomposed in terms of SQG and two-qubit ZZ interactions, as seen in section 2.1.

Now that we have the qubits and the couplings correctly arranged, we need to express every element in terms of the analog resource. In Galicia et al. this is explicitly constructed for a homogeneous analog Hamiltonian. Here, we will consider a realistic construction in which the couplings can take arbitrary values and signs, by following the same algorithm. At this point, we have to rewrite the evolution of our target NN Hamiltonian during a time t_f in terms of the analog source NN Hamiltonian. For this, we will employ $n - 1$ analog blocks of the source NN Hamiltonian, each of them running for a time t_j, $j = 1;\ldots;n - 1$. Between each block, we can change the sign of the couplings by applying X gates. A coupling where a X gate has been applied to one and only one of the qubits will evolve as if the time goes backwards. Putting all this together, we have the following equation that connects our target NN Hamiltonian with the source NN Hamiltonian blocks

$$
t_f H_{\mathrm{NN}} = \sum_{j=1}^{n-1} \sum_{\ell=1}^{n-1} t_j (-1)^{f_j(\ell)+f_j(\ell+1)} g_j Z_\ell Z_{\ell+1} = \sum_{j=1}^{n-1} t_j \Big(\prod_{k=1}^{n} X_j^{f_j(k)} \Big) H_{\text{NN source}} \Big(\prod_{k=1}^{n} X_j^{f_j(k)} \Big); \tag{21}
$$

where $f_j(k)$ is a binary function pointing that a X gate is applied to qubit k at step j if $f_j(k) = 1$ and g_j are the couplings between qubits $j \leftrightarrow j + 1$ of the source NN Hamiltonian.

Now, we have to design a scheme where some of the couplings change their sign, so that the evolution under these gates and the analog blocks corresponds to the evolution of the target NN Hamiltonian. For an odd step number (starting at 1), we have to apply a X gate every two qubits starting from the second one until $k = j - 1$, $f_j(2; 4; 6; :::; j - 1) = 1$. For an even step number, we have to apply it instead starting from the first qubit, $f_j(1; 3; 5; :::; j - 1) = 1$. If the time evolution for a block is negative, then we have to invert all interactions. To do this we have to apply a X gate every 2 qubits starting from the second qubit on top of the previously calculated gates. If by doing this two X gates are applied to the same qubit, they both cancel out, $X \cdot X = \mathbb{1}$. This scheme is codified into a matrix M of size $n - 1 \times n - 1$ that takes values 1 or -1. Since this matrix is invertible, i.e. $|M| \neq 0$, we can find a set of times t_j which reproduce the target couplings that we are seeking. More precisely, the time of evolution for each analog block can be obtained by solving the equation

$$\vec{b} = M\frac{\vec{\tau}}{t_f};\ b_j = \frac{g_j^0}{g_j}; \tag{22}$$

where g_j^0 are the couplings from the target NN Hamiltonian, $\vec{\tau}$ is the vector of t_j and M is a matrix of 1's in the diagonal and above and -1's below the diagonal

$$M = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 1 \\ -1 & 1 & 1 & \cdots & 1 & 1 & 1 \\ -1 & -1 & 1 & \cdots & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -1 & -1 & -1 & \cdots & 1 & 1 & 1 \\ -1 & -1 & -1 & \cdots & -1 & 1 & 1 \\ -1 & -1 & -1 & \cdots & -1 & -1 & 1 \end{pmatrix}. \tag{23}$$

The solution to this system of equations can be obtained as a closed expression:

$$\frac{t_j}{t_f} = \frac{b_j - b_{j-1}}{2} \tag{24}$$

for $j = 2; : : : ; n - 1$ and

$$\frac{t_1}{t_f} = \frac{b_1 + b_{n-1}}{2}. \tag{25}$$

Any solution that yields $t_j = 0$ would mean that it is not necessary to apply neither the j-th analog block nor the corresponding X gates.

If we were to use this paradigm to simulate a QA, we would first have to digitize the evolution into time independent steps. Then, each of the steps would be decomposed as explained in this section. In Fig. 7 the different layers of simulation and decomposition steps are depicted.

2.4 Digitized quantum annealing

Digitized quantum annealing (DQA) is the digitized version of QA to be run in either the DQC or the DAQC paradigms. The evolution of an analog time dependent system can be approximated by taking sufficiently small steps in which the Hamiltonian is considered time independent. In practice, the algorithm is based on the Trotter-Suzuki formula, which will be explained in more detail in the next section [30]. The evolution can be divided it into equally sized time steps,

$$U_{\text{annealing}} = \mathcal{T}\exp\left(\frac{i}{\hbar}\int_0^T dt H(t)\right) \quad U_{\text{discrete}} = \prod_{j=1}^{n-1}\exp\left(\frac{i}{\hbar}\Delta t H(j\,\Delta t)\right); \tag{26}$$

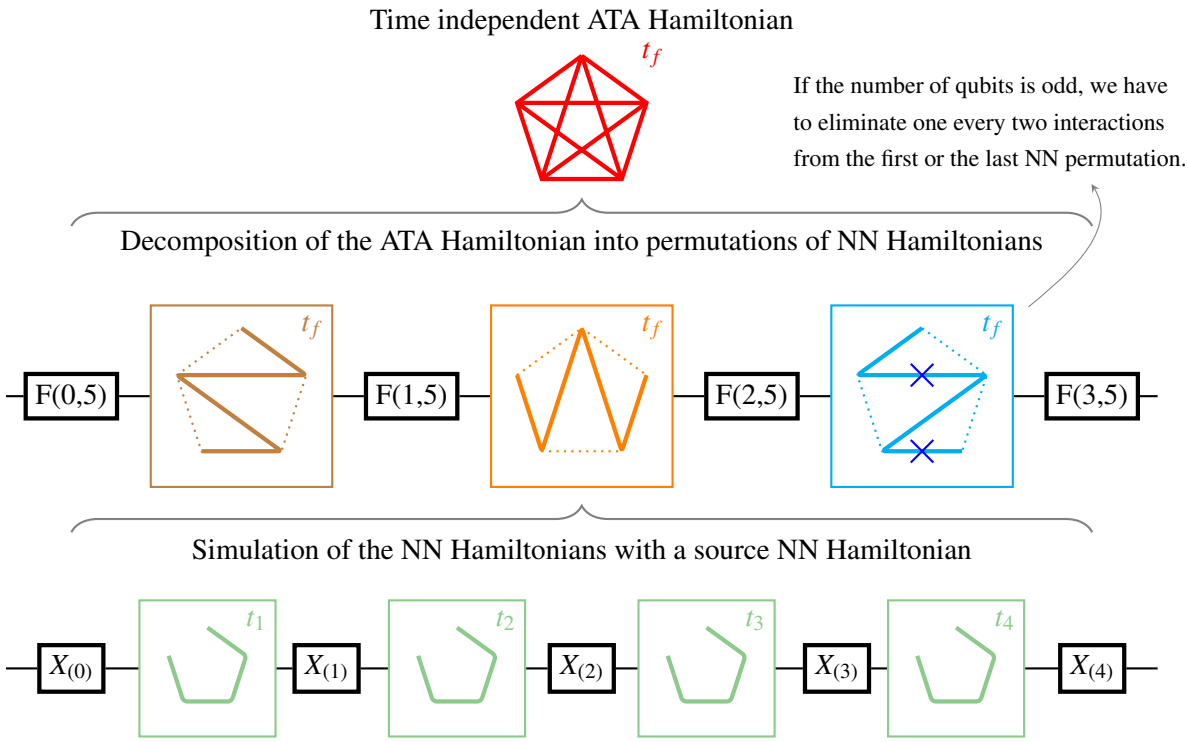


Fig. 7: Digital-Analog scheme to simulate the evolution of a time independent ATA Hamiltonian acting on n qubits for a time t_f, using for that a NN Hamiltonian as a source. The first step is to decompose the target ATA Hamiltonian into $\lceil N/2 \rceil$ NN Hamiltonians using the permutations from Eq. 19. The $F(k;n)$ blocks from Eq. 20 swap the qubit order to apply the said permutations. Finally, each of NN Hamiltonian is simulated using $i = 1;\dots;N-1$ analog blocks. Each of those analog blocks runs for a time t_j, calculated by solving Eq. 22. In between each analog block, X-gate blocks invert the sign of the effective interactions, matching the choice of the matrix M we made in Eq. 23.

where T is the total time of evolution and n is the number of steps into which the evolution is divided, so that $t = T/n$. As in the classical counterpart, n must be sufficiently large such that the Trotterization error is small and the approximation is as close as possible to the original operator.

Although this does not seem an improvement at first sight, the main advantage of this algorithm is that it can be implemented in a digital quantum computer. Also, when working in the DQC or DAQC, we have access to the advantages of these paradigms, such as the existence of error mitigation techniques. Additionally, his allows us to simulate any Hamiltonian with limited resources, even on SCC with low connectivity. The steps one have to follow to implement this algorithm are presented in algorithm 1.

Algorithm 1 Simulation of the evolution of an ATA Hamiltonian with DAQC

1: Prepare the system into the ground state of the Hamiltonian at time $t = 0$
2: Discretize the time evolution of the Hamiltonian $H(t)$ into n time steps of duration $t = T/n$
3: **for** each time step **do** . Simulate each of the time independent Hamiltonians $H(n\,t)$
4: Use the Trotterization of $\exp(-i\,tH(n\,t)/\hbar)$ to obtain an expression that is simulable using analog blocks
5: Simulate the analog blocks using the process described in section 2.3.1
6: Measure the system in the corresponding axes. For example, for a Ising Hamiltonian the measurement is made on the z axis

The main disadvantage of this algorithm is that it introduces two additional error sources when com-

pared with QA. On the one hand, the Trotterization of the time evolution introduces an error depending on t. On the other hand, the Trotterization of the time independent Hamiltonians also introduces an error, that depends on the order at which we truncate the expansion. In the next section these errors will be properly discussed.

2.5 Trotter-Suzuki formula

The Trotter-Suzuki formula gives an expression to decompose the exponential of a sum of operators as a product of the exponential of individual operators. The original formula was obtained in the context of group theory by Trotter [31]. The generalization of the formula and the expression to calculate the error committed by truncating the expansion at a finite order was obtained by Suzuki [32].

The Trotter-Suzuki formula gives a recursive definition for the approximants to the original exponential expression. The approximant of order m for the exponential of the sum of two operators A and B is constructed as

$$e^{A+B} \quad f_{n;m}(A;B) = \; e^{A=n} e^{B=n} e^{C_2=n^2} e^{C_3=n^3} \quad e^{C_m=n^m} \; {}^n; \tag{27}$$

where the operators C_i are recursively defined. The first two operators are

$$\begin{aligned} C_2 &= \frac{1}{2} \; \frac{@^2}{@ \; ^2} e \quad {}^B e \quad {}^A e \; {}^{(A+B)} \Big|_{=0} = \frac{1}{2}[B;A]; \\ C_3 &= \frac{1}{3!} \; \frac{@^3}{@ \; ^3} e \quad {}^{2C_2} e \quad {}^B e \quad {}^A e \; {}^{(A+B)} \Big|_{=0} = \frac{1}{3!}[C_2; A+2B]; \end{aligned} \tag{28}$$

and in general

$$C_m = \frac{1}{m!} \; \frac{@^m}{@ \; ^m} e \quad {}^{m \; 1 C_{m \; 1}} \quad e \quad {}^{2C_2} e \quad {}^B e \quad {}^A e \; {}^{(A+B)} \Big|_{=0} : \tag{29}$$

This expansion is exactly equal to the original exponential in the limit $n \; ! \; 1$. The error introduced by truncating at a finite m or n is provided the following theorem:

Theorem 2 (Trotter-Suzuki formula)**.** *For any two operators A and B in a Banach algebra,*

$$e^{A+B} \quad f_{n;m}(A;B) \quad \frac{c_{n;m}}{n^m (m+1)!} e^{\mathsf{k}A\mathsf{k}+\mathsf{k}B\mathsf{k}}; \tag{30}$$

where

$$0 \quad \lim_{n \; ! \; 1} c_{n;m} < 1; \tag{31}$$

$$c_{n;m} = (m+1)! n^{m+1} f_m \; \frac{1}{n} \; \exp\left(\sum_{k=2}^{0} \frac{\mathsf{k}C_k\mathsf{k}}{n^{k \; 1}} \right); \tag{32}$$

for $f_m(\;)$ constructed as the projection operator

$$f_m(\;) = \mathsf{P}_m \; \exp\left[\; 2 \; (\mathsf{k}A\mathsf{k} + \mathsf{k}B\mathsf{k}) + \; {}^2\mathsf{k}C_2\mathsf{k} + \quad + \; {}^m\mathsf{k}C_m\mathsf{k} \right]; \tag{33}$$

and where $\mathsf{P}_m(f(\;))$ is the sum of successive derivative terms

$$\mathsf{P}_m(f(\;)) = f(\;) \sum_{k=0}^{m} \frac{k}{k!} f^{(k)}(0): \tag{34}$$

For bounded operators A and B the approximant is equal to the exponential on the limit

$$\lim_{n \; ! \; 1} f_{n;m} = e^{A+B}: \tag{35}$$

We are interested in calculating the error of truncating the Trotterization at first order, $m = 1$ and for $n = 1$. We can apply here the result from the theorem. Then, we have that the first order approximant is $f_{1;1}(A; B) = e^A e^B$, and the truncation error is

$$e^{A+B} \quad e^A e^B \quad \frac{c_{1;1}}{2} e^{\|A\|+\|B\|}; \tag{36}$$

where

$$c_{1;1} = 2 f_1(1) \exp\left(\frac{\|[B;A]\|}{2}\right); \tag{37}$$

and

$$f_1(1) = P_1 \left(\exp\left(2(\|A\| + \|B\|) + \frac{\|[B;A]\|}{2}\right)\right) = \exp\left(2(\|A\| + \|B\|) + \frac{\|[B;A]\|}{2}\right) \quad 1 \quad 2(\|A\| + \|B\|): \tag{38}$$

We can now repeat the calculation, but for a constant multiplying both operators, as this is the case for the evolution operator. We want to approximate $e^{t(A+B)}$ at first order. The approximant for this is $e^{tA} e^{tB}$ and the error is bounded in this case by

$$e^{t(A+B)} \quad e^{tA} e^{tB} \quad \frac{1}{2} \left(\exp\left(2t(\|A\| + \|B\|) + t^2 \frac{\|[B;A]\|}{2}\right)\right) \quad 1 \quad 2t(\|A\| + \|B\|) \left(\exp\left(t^2 \frac{\|[B;A]\|}{2}\right)\right) e^{t(\|A\|+\|B\|)}. \tag{39}$$

We observe that in this case the error grows as $O(\exp(t^2 \|[B;A]\|))$.

We can improve this result without employing higher order terms. The symmetrized Trotter-Suzuki formula [33] for the first order reduces the error by using the following approximant

$$e^{t(A+B)} \quad e^{tA=2} e^{tB} e^{tA=2}; \tag{40}$$

where the positions of the operators can be switched without changing the result. The order of magnitude of this error is $0(t^3)$, which is a huge improvement considering that we have just added one more term to the approximant.

This formula can be extended for the exponential of an arbitrary number of operators. In this case, we have a more general expression.

Theorem 3 (Trotter-Suzuki formula)**.** *For an arbitrary number p of operators A_j in a Banach algebra*

$$\exp\left(\sum_{j=1} A_j\right) \quad f_{n;m}(\{A_j\}) \quad \frac{c_{n;m}}{n^m (m+1)!} \exp\left(\sum_{j=1} \|A_j\|\right); \tag{41}$$

where the approximant is

$$f_{n;m}(\{A_j\}) = \left(e^{A_1=n} \quad e^{A_p=n} e^{C_2=n^2} \quad e^{C_m=n^m}\right)^n; \tag{42}$$

$c_{n;m}$ is given by Eq. 32 and

$$f_m(\ ) = P_m \left(\exp\left(2 \sum_{j=1} \|A_j\| + {}^2\|C_2\| + \quad + {}^m\|C_m\|\right)\right): \tag{43}$$

If all operators $\{A_j\}$ are bounded, then

$$\lim_{n \to 1} f_{n;m}(\{A_j\}) = \exp\left(\sum_{j=1} A_j\right): \tag{44}$$

Taking into account that Hilbert spaces are also Banach spaces, these formulae work for quantum mechanics with its usual L^2 or Frobenius norm. Furthermore, as quantum gates and evolution operators are unitary, an thus bounded, Eqs. 35 and 44 always hold.

The formulae can be extended to time evolution operators with a time-dependent Hamiltonian into steps of evolution with time-independent Hamiltonians. For the particular case of time steps with equal length, the decomposition is [30]

$$U(T) = \mathsf{T} \exp\left(i \int_0^T H(t)dt \right) \prod_{j=0}^{n_T - 1} e^{-iH(j\,\delta t)\delta t/\hbar}; \tag{45}$$

where T is the time ordering operator and n_T is the number of steps of length $\delta t = T/n_T$. Applying the Suzuki-Trotter formula, we see that for infinitely many steps ($n_T \to \infty$) this expansion is exact at any order m.

Lets take now the case of a Hamiltonian that can be separated into two non commuting terms, $H(t) = H_1(t) + H_2(t)$ with $[H_1(t), H_2(t)] \neq 0$. Note that this form is the same as the Hamiltonian used for QA in Eq. 14, in particular it has the form of the Ising Hamiltonian from Eq. 18. Suppose a finite number of steps n_T and the Trotterization truncated at first order. Then, the Trotterization error of each step can be estimated by the expression [34],

$$U(t;t+\delta t) \approx e^{-i\int_t^{t+\delta t} dt H_1(t)/\hbar} e^{-i\int_t^{t+\delta t} dt H_2(t)/\hbar} \quad \frac{c_{12}\,\delta t^2}{\hbar^2} \quad \frac{c_{max}^2\,\delta t^2}{2\hbar^2}; \tag{46}$$

with

$$c_{12} = \frac{1}{\delta t^2} \int_t^{t+\delta t} dx \int_t^x dy \|[H_1(y), H_2(x)]\|; \tag{47}$$

and

$$c_{max} = \max_t \max_{t+\delta t} \max_{i=1,2} \|H_i(\tau)\|. \tag{48}$$

From this expression, we can obtain the error introduced in the process of separating the evolution using the Trotter formula at first order. For this, we can calculate the upper bound of this error by taking the maximum error for a single step and multiplying it by the number of steps,

$$U(T) \approx \prod_{j=0}^{n_T - 1} e^{-i\int_{j\,\delta t}^{(j+1)\,\delta t} dt H_1(t)/\hbar} e^{-i\int_{j\,\delta t}^{(j+1)\,\delta t} dt H_2(j\,\delta t)/\hbar} \quad n_T \frac{c_{max}^2\,\delta t^2}{2\hbar^2}; \tag{49}$$

where now c_{max} is calculated on the range from 0 to T.

3. LND problem and previous approaches

In this chapter, we explain the LND problem, which we address with our algorithm in the following chapter. We provide the mathematical formulation, and we relate it with the real life problem it tries to solve. Then, we briefly review the previous work addressing the LND problem using classical algorithms, more specifically, metaheuristic algorithms used to solve it approximately. We end this chapter by explaining a previous work which solves the problem by using an hybrid classical-quantum algorithm.

3.1 LND problem definition

The LND problem can be defined as the problem of finding the optimal network that connects manufacturers to its clients through a series of supply lines achieving the minimum possible cost. This problem can be stated in a large amount of di erent ways, with varying amount of layers, constraints and complexity levels. These problems may take into account multiple sources of cost depending on the problem, such as factory building cost, transport cost, production cost and a large etc. As the problem formulation can grow indefinitely given a real life scenario, a good algorithm to solve this problem must be capable of adapting to this changes.

The main objective of this master thesis is to solve a simple version the LND problem. In order to keep the number of variables and constraints at reasonable levels, we will focus on the version of the problem used in Ding et al. [35]. This LND problem consists on finding the best factory and supply line configuration, in a network that directly connects I clients with a set of J factories, where the graph representing the connections is a complete bipartite graph as shown in Fig. 8. The cost of the network is given by the cost of constructing the factories f_j and the transport costs of each of the active supply lines c_{ij} connecting the client $i = 1; :::; I$ to the factory $j = 1; :::; J$. The optimal configuration is then given by a vector containing the configuration of the built factories x_j and a matrix representing the active supply lines y_{ij}. The total cost of the logistic network is given by

$$F_{\text{cost}}(x; y) = \sum_j f_j x_j + \sum_i \sum_j c_{ij} y_{ij}: \tag{50}$$

Now we have to add some limitations and conditions to the solutions we want to obtain.

First, all the clients must receive the products. Thus, each client must be connected to at least one of the factories through an active supply line. In this version of the problem, the condition is even more strict, each client must be connected to one and only one of the factories. This imposes a first constraint on the allowed configurations, so that the solution to the problem must fulfil the condition

$$\sum_j y_{ij} = 1; \ 8i: \tag{51}$$

Second, the clients might have di erent needs. Thus, each of the customers demand a certain amount of goods from the factories, specified with the variable d_i. Also, it is reasonable to think that each factory have a limit on how many goods they can produce, given by the variable v_j. With this two variables we can set an obvious limitation, a factory cannot supply more than it produces. Although in real life scenarios the production of a factory could be stretched if needed, the formulation given here does not contemplate this option. This being the case, the constraint can be expressed as

$$\sum_i d_i y_{ij} \quad v_j; \ 8j: \tag{52}$$

Lastly, we have a clear constraint on the allowed configurations. If a factory is not built, a supply line can not depart from it. This last constraint is given as

$$y_{ij} \quad x_j; \ 8i; \ 8j: \tag{53}$$

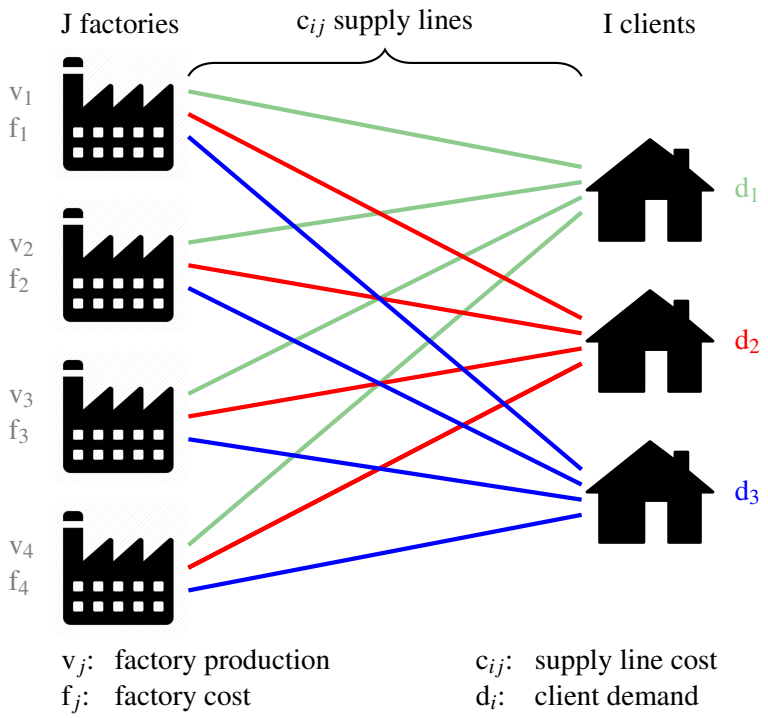


Fig. 8: Depiction of the LND problem for 4 factories and 3 clients. The bipartite graph connecting the factories and the clients is present here, with a total of 12 di erent supply lines. The solution to the problem consists on only one supply line per client, labeled with di erent colors, and a selection of built factories. A solution for the problem can not have a supply line departing from a non built factory, a factory dispatching more goods than it produces, nor a client receiving less goods than the demanded.

Taking into account only the unconstrained combinations of the binary variables x_j and y_{ij}, the total amount of possible configurations is $2^{J(I+1)}$. However, the first and the third constraints always allows the same solutions, as they don't depend on the input parameters. The number of allowed configurations after applying this two constraints is given by

$$\mathrm{comb}(I;J) = \sum_{n=1}^{J} \binom{J}{n} n^I. \tag{54}$$

Proof. To prove the this expression we will count the number of states that both of the constraints discard at the same time. Let us first think about a problem with only one client. As the number of active supply lines per client must be exactly one, the number of possible di erent solutions is equal to the number of active factories, n. It is evident that $1 \quad n \quad J$, since a solution where no factories are built is never allowed. The next step then is to add more clients to the problem. As we are not taking into account the second constraint, the choice of one factory for a client does not interfere with the choice of the rest. Indeed, we could have a built factory not sending products to any client. For 2 clients, the combination of two independent sets of n configurations yields a total of n^2 possible configurations. From this point, we deduce that for I clients there are a total of n^I combinations. On top of the possible client configuration for a set n, the factories also have di erent configurations, which don't have any constraint at all. The number of these configurations for n factories is given by the binomial coe cient of J choosing n. For each n factory combination, we have n^I total supply line configurations, so the total number of possible solutions would be the sum over $1 \quad n \quad J$ of J choose n factory configurations with n^I supply line combinations each. □

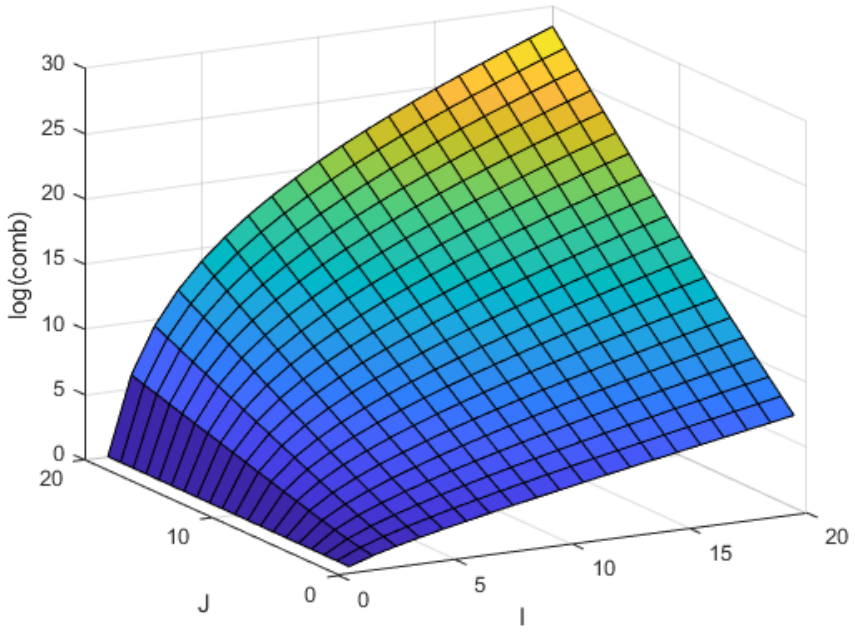


Fig. 9: Logarithmic plot of the number of allowed solutions after applying the first (Eq. 51) and the third constraint (Eq. 53) in terms of the number of factories J and the number of clients I of the problem, i.e. the logarithmic plot of Eq. 54.

We can calculate the number the allowed configurations and plot it in terms of the number of factories and clients, as we show in Fig. 9. Here we see that the number of allowed combinations grows exponentially with both the number of factories and clients. This reinforces the claim that this problem is NP-Hard.

The second restriction depends on the input parameters of the problem. If they are too restrictive, the amount of allowed configurations could be drastically reduced. If this was the case, a simple search algorithm could give the exact solution for this problem in a reasonable number of steps. Unfortunately, this information cannot be deduced a priori. In fact, in order to obtain the set of allowed states one would need to check all the configurations.

Taking this into account, a naive exact algorithm for solving this problem is shown in algorithm 2. Suppose that there is an algorithm to obtain the set of configurations allowed by the first and third constraints S in just one step, $O(1)$. Even in that situation, the complexity of the algorithm still grows as the expression obtained in Eq. 54.

Algorithm 2 Exact algorithm to solve the LND problem

Require: Costs f_j and c_{ij}, constraint parameters d_i and v_j

```
Generate the set of configurations allowed by the first and third constraints S
for each S_n from S do
    if S_n is allowed by the second constraint then
        Calculate the cost of S_n, F_cost(S_n)
    else
        Delete S_n from the list
Search the S_n that minimizes the cost
```

The exponential cost of this exact algorithm makes it not feasible for large problems. To solve the problem in a reasonable number of steps, we need to search for other kind of algorithms. For this kind

of tasks, metaheuristic algorithms gives approximate solutions to hard problems. We will present some of them in the next section.

3.2 Classical algorithms for the LND problem

The LND problem has been extensively studied due to its industrial and economical applications. Most of the algorithms developed to solve this problem are metaheuristic, which applies an strategy in order to reduce the number of steps to try to find an approximate solution to the problem. An usual condition for these algorithms is that they must converge to the exact solution after an infinite number of steps. As the bibliography discussing these algorithms is vast and it is out of the scope of this thesis to review it, we will just mention a couple of the most interesting ones. Metaheuristic algorithms can be divided into various groups, depending on the strategies they implement or the source of inspiration to develop them. On the realm of evolutionary algorithms we find genetic algorithms (GA) [36], hybrid versions of GA [37] or hybrid evolutionary algorithms [38]. Bio-inspired swarm algorithms have also been used to solve the LND problem, among them: ant colony optimization [39], artificial bee colony [40] or particle swarm [41]. Other examples of various algorithms are the chaotic optimization [42], simulated annealing [43] or surrogate model based optimization [44]. We will introduce some basic ideas of two of the most interesting algorithms, GA and ACO.

GA is based on the crossover and mutation of genes that occurs during a evolutionary process. In this case, the objective is to optimize the solution of the LND problem. As the input parameters for the cost function is a binary array, we can directly assign them as the genes for an individual. The algorithm starts with a pool of randomly generated individuals. In each iteration, these individuals compete with each other to try to pass its genes to the next generation. The individuals that participates in the crossover gets selected according to their fitness value, prioritising the best individuals. Among the di erent ways of mixing the genes of the individuals, the simplest one is the single point crossover, where the first genes of one individual are concatenated with the last genes of the other, and where the cutting point is randomly chosen. Since this the crossover strategy is random, there is a risk of loosing the best individual so far. To avoid that, the elitism strategy takes the best individuals and pass them directly to the next generation. Finally, the mutation strategy takes some of the individuals and changes some of their genes at random, ensuring a homogeneous exploration of solutions. The algorithm then runs for a certain amount of iterations or until no better solutions are found.

This paragraph is a free interpretation of Silvia et al. [39], since their LND problem has more layers than the one presented in this thesis.ACO takes the behaviour of an ant colony to find the shortest path from the nest to a source of food. Instead of communicating with each other directly, the ants leaves a pheromone trail behind them, so that the next ants can follow it. For the LND problem we are solving in this thesis, the ants starts on a random client. From there, they have to chose one of the paths to traverse to arrive to a factory. Given a exploration probability, the ants can decide whether to chose the factory randomly or to follow the pheromone trail left by the previous generations. When the ants are guided by the pheromones, the probability of choosing a path is given by the pheromone distribution along all accessible paths. When the ant arrives the factory, it takes the goods from the factory that corresponds with the demand of the client from where they started. If they can deliver back the goods, the ant deposits a pheromone trail along the trail. If they couldn't bring enough goods, then the path is penalized and no trail is left. After all the clients had an ant trying to deliver the goods, the amount of pheromones deposited in the traversed supply lines is calculated according to the cost function, depositing more pheromones the lower the cost is. An important strategy to avoid converging to a local minimum is the evaporation strategy, which consists on subtracting a certain percentage of the pheromones from all paths. Then again, the process is iterated a certain amount of times or until the algorithm has converged to some result.

It is extremely di cult to compare the e ectiveness of the di erent algorithms, as they are usually optimized to solve a specific kind of problem. This means that we do not have a common benchmark

to test these algorithms. If we wanted to see what are the best metaheuristic algorithm, first we would have to rewrite them in terms of the problem presented at the start of this section. Then we would have to find the optimal input parameters and, at the end, we would have to run the benchmark on problems of di erent sizes. This work is not part of the thesis, so that we will not make any further analysis in this regard.

3.3 Hybrid classical-quantum annealing algorithm for the LND problem

The algorithm we develop is heavily based on the hybrid classical-quantum annealing algorithm developed recently by Ding et al. [35]. As noted before, in that work they proposed an algorithm solving the same problem as the one we are solving. Their algorithm consist on dividing the problem into two layers. The outer layer takes the factory configurations, while the inner layer comprises the supply chain configuration according to the outer layer. The algorithm first generates a new factory configuration with a classic algorithm, and then it solves the supply line optimization problem using a a quantum annealing process.

The outer layer algorithm to generate a new configuration consist on finding a new vector $\tilde{x}$ that is close to the previous one x. For this, the algorithm has three strategies:

1. Close a built factory: pick a random j among all the positions where a factory is built $x_j = 1$ and change its value to $\tilde{x}_j = 0$.
2. Build a new factory: similar to the previous case, pick a random j among all the positions where there is no factory $x_j = 0$ and change its value to $\tilde{x}_j = 1$.
3. Move a factory: choose a random factory $x_j = 1$ and a random empty place $x_{j'} = 0$ and swap their values, $\tilde{x}_j = 0$ and $\tilde{x}_{j'} = 1$.

In each iteration, one of the three operations is randomly chosen and applied once. Note that in the case in which all factories (no factories) are built, the first (second) operation is not allowed.

After a new factory configuration is generated, the inner layer is solved by a quantum annealing process. The algorithm is designed to be run on a D-Wave's annealer. This imposes a restriction on what kind of problems can be solved, specifically quadratic unconstrained binary optimization (QUBO) problems. QUBO problems can be stated as finding the binary input vector x that minimizes the cost function

$$F(x) = xAx^t + xB \quad \text{constraints}; \tag{55}$$

where A is a square matrix and B a vector. The problem is encoded into A and B, along the restrictions imposed by the constraints.

Before obtaining the Hamiltonian for the annealing process, the constraints must me rewritten so that they can be introduced in the problem as an extra term in the QUBO formulation. For this version of the hybrid QA only the first two constrains are taken into account. For the constraint expressed as an inequality, they introduce slack variables in the form of a binary expansion, $\hbar 2; a_j \mathrm{i} = \sum_{k=0} 2^{\backprime} a_{j^{\backprime}}$, as seen at the end of section 2.2. The binary expansion introduces the new binary variables $a_{j^{\backprime}}$, optimizing the number of qubits needed to implement the Hamiltonian. With both constraints written as equalities, the variables can be written in one side of the equations and then squared. This way, the terms that penalize the violations of the first two constraints can be written as

$$\left(\sum_j y_{ij} \quad 1 \right)^2 ; \tag{56}$$

and

$$\left(\sum_i d_i y_{ij} + \hbar 2; a_j \mathrm{i} \quad v_j \right)^2 : \tag{57}$$

The penalty introduced by these two expressions grows with the square of the Hamming distance that separates a configuration from a non allowed one. To illustrate this, we can think of two states which are not allowed by the first constraint: one has no factories linked to a client, and the other one has three factories supplying a single client. The first case is in a 1 bit flip distance to an allowed solutions with a penalty of 1, while the second is in a 2 bit flip distance with a penalty of 4. This strategy highly penalizes the states that violates the constraints by far, making the system converge to allowed solutions faster than with a linear penalty system.

The penalties must introduce enough extra cost (energy) to the configuration (state) so that they have always more cost than the optimal configuration. In their algorithm, they aim to make all the prohibited configurations to have much worse cost than any of the allowed configurations. To do this, they multiply the expressions from equations 56 and 57 by the penalty factors $_i$ and $_j$ respectively. The strategy they use to choose the penalties is to set them as $_i \quad _j \quad c_{ij}$, making the first condition more heavily enforced than the second one.

Although not applied to the algorithm, the article mentions the possibility of introducing a penalty term for the last constraint. This could be accomplished by introducing binary slack variables z_{ij} with a term that would take into account both the supply line and the factory variables (x_j). The penalty term proposed is of the form $(x_j \quad y_{ij} \quad z_{ij})^2$, where would be the penalty factor. However, as the factory variables are already on the outer layer of the problem, they are not introduced into the annealing process.

Now, the problem Hamiltonian for the annealing is written summing the cost function for the supply lines and the constraints,

$$H_P = c_{ij}\hat{y}_{ij} + {}_i(\hat{y}_{ij} \quad \mathbb{1})^2 + {}_j(d_i y_{ij} + 2\grave{}\hat{a}\cdot_j \quad v_j)^2; \tag{58}$$

where the operators with the hats must be transformed to $\hat{q} \; ! \; (\mathbb{1} \quad {}^z)=2$ as anticipated in section 2.2. Here we also have introduced a sort of Einstein's notation, where the terms are to be sum over matching subindices. This Hamiltonian is introduced in place of the H_1 term from Eq. 14. As it has the same form as the Ising Hamiltonian, it is compatible with the D-Wave quantum annealers. The annealing process starts with initial state of each qubit in the ground state of the ^{x} Pauli operator, $(\mathrm{j1i} \quad \mathrm{j0i})=\sqrt{2}$, which can be generated by applying a Hadamard gate followed by a Z gate to every qubit. The final state is measured only in the z axis of the qubits where the information of the configuration is stored, i.e. the qubits corresponding to y_{ij}.

With the inner layer algorithm explained, the full algorithm to solve the LND problem is based on a simulated annealing algorithm, in which each iteration first solves the outer and then the inner layers. The newly generated solution is then accepted by applying the Metropolis algorithm, if the solution is better then the previous one then it is always accepted. Else, it is only accepted with a probability given by the acceptance ratio, which allows only better solutions to be accepted as the temperature gets colder. After each step, the annealing temperature adjusts following the cooling rate, the same way as in a classical simulated annealing algorithm. These steps are wrapped-up in algorithm 3.

Algorithm 3 Hybrid classical-quantum annealing algorithm for LND

Require: Costs f_j and c_{ij}, constraint parameters d_i and v_j, penalty variables $_i$ and $_j$, annealing parameters initial T_i and final temperatures T_f and cooling rate, maximum iterations

Initialization of SA temperature $T = T_i$, factory configuration $x_j = 0$ and set the current solution cost to infinity, $cost(x_j; y_{ij}) = 1$
while $T < T_F$ **do**
  for maximum number of iterations **do**
    Generate a new factory configuration, $\tilde{x}_j$
    Perform the quantum annealing process to solve the inner layer, $\tilde{y}_{ij}$
    if $cost(x_j; y_{ij}) > cost(\tilde{x}_j; \tilde{y}_{ij})$ **then** . Metropolis algorithm for solution acceptance
      Accept the new configuration as the solution, $x_j = \tilde{x}_j$, $y_{ij} = \tilde{y}_{ij}$
    else
      Generate a random number, 0 1
      if $<$ exp[$(cost(x_j; y_{ij})$ $cost(\tilde{x}_j; \tilde{y}_{ij}))/T$] **then**
        Accept the new configuration as the solution, $x_j = \tilde{x}_j$, $y_{ij} = \tilde{y}_{ij}$
  Adjust the annealing temperature according to the cooling rate
return current configuration x_j, y_{ij}

4. Digitized QA for LND

In this section we present the digitized QA algorithm for LND, which can be framed in the DAQC paradigm. Compared to the algorithm it is based on, discussed in the previous section (sec. 3.3), this digitized version is suitable to be implemented in any digital quantum computer. On top of this, the algorithm codifies all the constraints of the problem into the annealing Hamiltonian, allowing it to be implemented as a full quantum algorithm. Also, as we will show later, the better choice of the penalty parameters makes it faster in terms of the time needed to assure the adiabaticity of the process.

This section is structured in the following way. We start by presenting the specific codification we used both for the problem (sec. 4.1) and its constraints (sec. 4.2). We discuss the proposed selection for Hamiltonian parameters, in particular, we give a new strategy to choose the penalty factors applied to the invalid solutions (sec. 4.3). We make some comments about the annealing schedule and the digitization of the time evolution (sec. 4.4). To end the discussion about the algorithm itself, we wrap up all the steps prior to this point to give an explicit expression for the evolution of the system (sec. 4.5), and we define the initial state and the measurement of the final state (sec. 4.6). In section 4.7, we explore the topology of the SCC in which the algorithm could be implemented. We also comment the source of errors of our algorithm and their e ect on the fidelity of the solution (sec. 4.8). To end this chapter we extend the initial LND problem giving some guidelines for introducing new cost functions and constraints (sec. 4.9). Throughout this chapter, we elaborate some of the steps to solve an small LND problem (Exs. 1-4).

4.1 Codification of the LND problem

The codification used in this algorithm is exactly the same as the one referenced in section 3.3. The factory and supply line configuration is given by a vector and matrix of binary variables x_j and y_{ij} respectively. We have to transform the binary variables to the eigenstates of the Pauli z operators described in section 2.2. This transformation yields the new variables $x_j \to \hat{z}_j=2$ and $y_{ij} \to \hat{z}_{ij}=2$, where the value 1 (0) corresponds to the excited (ground) state $|1\rangle$ ($|0\rangle$). Here, the subindices of the Pauli operators label each of the qubits of the system.

To codify the problem parameters, i.e. the factory f_j and supply line costs c_{ij}, we introduce them in the Hamiltonian as multiplicative terms. This way, we can write the Hamiltonian of the cost of the logistic network as

$$H_{\text{cost}} = f_j\hat{x}_j + c_{ij}\hat{y}_{ij} = f_j\frac{\mathbb{1} \quad \hat{z}_j}{2} + c_{ij}\frac{\mathbb{1} \quad \hat{z}_{ij}}{2} = \frac{f_j}{2}\hat{z}_j \quad \frac{c_{ij}}{2}\hat{z}_{ij} + E_0: \tag{59}$$

Note here that as the origin of the energy scale is at an arbitrary point, we can ignore any energy constants. This will be systematically done from this point on.

4.2 Constraint codification

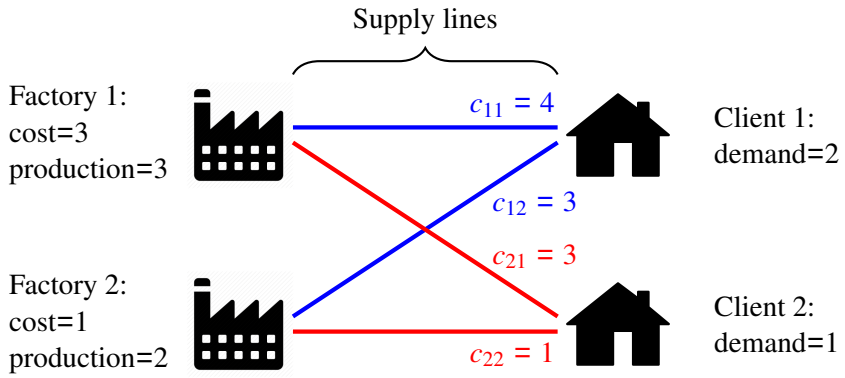
For the constraint codification we will take the expressions 56 and 57 from Ding et al. [35]. But in contrast to that algorithm, in which the third constraint couldn't be introduced in the Hamiltonian, now we can. However, we will not use the codification they proposed. Instead, we use an optimal codification for introducing the less ancillary qubits as possible. For that, we can use the binary codification to introduce the corresponding slack variables. Doing this, we can write each of the constraint as the following penalty terms:

$$\begin{aligned} H &= \sum_i \hat{y}_{ij} \quad 1\Big)^2 = \sum_i \hat{y}_{ij}^2 + 2\hat{y}_{ij1}\hat{y}_{ij2} \quad 2\hat{y}_{ij} + \mathbb{1} = \sum_i \quad \hat{z}_{ij} + 2 \quad \hat{z}_{ij1}\hat{z}_{ij2} \quad \hat{z}_{ij1} \quad \hat{z}_{ij2} + 2\hat{z}_{ij} \\ &= \sum_i 2\hat{z}_{ij1}\hat{z}_{ij2} + (3 \quad 2J)\hat{z}_{ij}; \end{aligned} \tag{60}$$

Example 1: Toy problem and its codification

We illustrate how to obtain the solution to a particular problem using the proposed algorithm using a toy problem. The problem has a small number of clients and factories, but enough have insight of the algorithm. Along this chapter, we will expand some of the calculations needed to implement it.

The problem is set for 2 clients demanding $d = [2;\ 1]$ and 2 factories producing $v = [3;\ 2]$ with a cost of $f = [3;\ 1]$. The supply lines connecting them have an associated transport cost of $c = [4;\ 3;3;\ 1]$. These parameters are chosen so that the the cheapest configuration allowed by the first and third constraints is not allowed by the second. Also, the cost difference between the rest of the allowed solutions is small, making it for the algorithm harder to escape from these local minima. This forces the algorithm to implement all the constraints, making this example illustrative of all the strategies. To obtain the solution for this particular problem is trivial, as we can just calculate all the few possible configurations and calculate the solution. The optimal configuration takes the goods from the first factory to the second client and vice-versa, with a total cost of 9.



With the problem stated, we will give now the codification of the cost function on the Hamiltonian. Taking the expression from Eq. 59 and filling in the parameters, we obtain the corresponding Hamiltonian,

$$H^{(toy)}_{\text{cost}} = \frac{3}{2}\,\hat{z}_1 \quad \frac{1}{2}\,\hat{z}_2 \quad 2\,\hat{z}_{11} \quad \frac{3}{2}\,\hat{z}_{12} \quad \frac{3}{2}\,\hat{z}_{21} \quad \frac{1}{2}\,\hat{z}_{22}.$$

It is easy to check the energy of the solution for the problem. The terms being all local, the solution is $|1_{j1};1_{j2};1_{i1\,j1};0_{i1\,j2};0_{i2\,j1};1_{i2\,j2}\rangle$, with an energy of 3=2. The rest of states allowed by the first and third constraints are $|1_{j1};0_{j2};1_{i1\,j1};0_{i1\,j2};1_{i2\,j1};0_{i2\,j2}\rangle$ and $|1_{j1};1_{j2};0_{i1\,j1};1_{i1\,j2};1_{i2\,j1};0_{i2\,j2}\rangle$ with an energy of 5=2 and $|0_{j1};1_{j2};0_{i1\,j1};1_{i1\,j2};0_{i2\,j1};1_{i2\,j2}\rangle$ with a energy of 0.

$$\begin{aligned}
H &= \sum_j \Big(d_i\hat{y}_{ij} + 2^k\hat{q}_{kj} \quad v_j\mathbb{1} \Big)^2 \\
&= \sum_j \Big(d_i^2\hat{y}_{ij}^2 + 2d_{i1}d_{i2}\hat{y}_{i1\,j}\hat{y}_{i2\,j} + 2^{k+1}d_i\hat{y}_{ij}\hat{q}_{kj} \quad 2v_jd_i\hat{y}_{ij} + 2^{2k}\hat{q}_{kj}^2 + 2^{k1+k2+1}\hat{q}_{k1\,j}\hat{q}_{k2\,j} \quad 2^{k+1}v_j\hat{q}_{kj} + v_j^2\mathbb{1} \Big) \\
&= \sum_j \Big(\frac{d_i^2 \quad 2d_iv_j}{2}\,\hat{z}_{ij} \quad \frac{2^{2k} \quad 2^{k+1}v_j}{2}\,\hat{z}_{kj} + \frac{2d_{i1}d_{i2}}{4}\,\hat{z}_{i1\,j}\,\hat{z}_{i2\,j} \quad \hat{z}_{i1\,j} \quad \hat{z}_{i2\,j} \\
&\qquad + \frac{2^{k1+k2+1}}{4}\,\hat{z}_{k1\,j}\,\hat{z}_{k2\,j} \quad \hat{z}_{k1\,j} \quad \hat{z}_{k2\,j} + \frac{2^{k+1}d_i}{4}\,\hat{z}_{ij}\,\hat{z}_{kj} \quad \hat{z}_{ij} \quad \hat{z}_{kj} \Big) \\
&= \sum_j \Big(\frac{d_i}{2}\Big(\sum_i d_i \quad 2v_j + 2^{K_j} \quad 1\Big)\,\hat{z}_{ij} \quad 2^k \quad 1\Big(\sum_i d_i \quad 2v_j + 2^{K_j} \quad 1\Big)\,\hat{z}_{kj} + \frac{d_{i1}d_{i2}}{2}\,\hat{z}_{i1\,j}\,\hat{z}_{i2\,j} \\
&\qquad + 2^{k1+k2} \quad 1\,\hat{z}_{k1\,j}\,\hat{z}_{k2\,j} + 2^k \quad 1\,d_i\,\hat{z}_{ij}\,\hat{z}_{kj} \Big);
\end{aligned} \tag{61}$$

where $k = 0; \dots; K_j - 1$ and $k1 < k2$ with $k1; k2 \geq k$, and

$$
\begin{aligned}
H &= \sum_j \Big(I\hat{x}_j - \hat{y}_{ij} - 2^{\grave{}}\hat{q}_{\grave{}j} \Big)^2 \\
&= \sum_j \Big(I^2\hat{x}_j^2 - 2I\hat{x}_j\hat{y}_{ij} - 2^{\grave{}+1}I\hat{x}_j\hat{q}_{\grave{}j} + \hat{y}_{ij}^2 + 2\hat{y}_{i1j}\hat{y}_{i2j} + 2^{\grave{}+1}\hat{y}_{ij}\hat{q}_{\grave{}j} + 2^{2\grave{}}\hat{q}_{\grave{}j}^2 + 2^{\grave{}1+\grave{}2+1}\hat{q}_{\grave{}1j}\hat{q}_{\grave{}2j} \Big) \\
&= \sum_j \Big(\frac{I^2}{2} - \frac{z}{j} - \frac{2I}{4} - \frac{z}{j}\frac{z}{ij} - \frac{z}{j} - \frac{z}{ij} - \frac{2^{\grave{}+1}I}{4} - \frac{z}{j}\frac{z}{\grave{}j} - \frac{z}{j} - \frac{z}{\grave{}j} - \frac{1}{2} - \frac{z}{ij} - \frac{2^{2\grave{}}}{2} - \frac{z}{\grave{}j} \\
&\quad + \frac{2}{4} - \frac{z}{i1j}\frac{z}{i2j} - \frac{z}{i1j} - \frac{z}{i2j} + \frac{2^{\grave{}+1}}{4} - \frac{z}{ij}\frac{z}{\grave{}j} - \frac{z}{ij} - \frac{z}{\grave{}j} + \frac{2^{\grave{}1+\grave{}2+1}}{4} - \frac{z}{\grave{}1j}\frac{z}{\grave{}2j} - \frac{z}{\grave{}1j} - \frac{z}{\grave{}2j} \Big) \\
&= \sum_j \Big(\frac{I(2^L - 1)}{2} - \frac{z}{j} - \frac{2^L - 1}{2} - \frac{z}{ij} - 2^{\grave{}-1}(2^L - 1) - \frac{z}{\grave{}j} - \frac{I}{2} - \frac{z}{j}\frac{z}{ij} - 2^{\grave{}-1}I - \frac{z}{j}\frac{z}{\grave{}j} - \frac{1}{2} - \frac{z}{i1j}\frac{z}{i2j} \\
&\quad + 2^{\grave{}-1} - \frac{z}{ij}\frac{z}{\grave{}j} + 2^{\grave{}1+\grave{}2-1} - \frac{z}{\grave{}1j}\frac{z}{\grave{}2j} \Big);
\end{aligned}
\tag{62}
$$

where $\grave{} = 0; \dots; L - 1$ and $\grave{}1 < \grave{}2$ with $\grave{}1; \grave{}2 \geq \grave{}$. The slack variables are labeled with $\hat{q}$, and $_i$, $_j$ and $_j$ are the penalty factors for each respective constraint. Some of the factors appearing on the Hamiltonian terms come from the sum over repeated terms.

The total amount of ancilla qubits needed to code the slack variables depends on the size and the input parameters of the problem. For the second constraint, we need to introduce $K_j = \lceil \log_2(v_j + 1) \rceil$ qubits. This means we need a di erent number of ancilla qubits for each $j = 1; \dots; J$. In the case of the third constraint, the number of qubits for each di erent j is constant, $L = \lceil \log_2(I + 1) \rceil$. This codification is more optimal than the codification proposed in the reference, which proposes to use IJ ancilla qubits to implement the third constraint. To sum up, for this algorithm we use $IJ + J$ qubits to code the solutions plus $\sum_j K_j + JL$ ancilla qubits, for a total of $N = J(I + L + 1) + \sum_j K_j$ qubits.

4.3 Hamiltonian parameter selection

The election of the penalty factors and other free parameters for this algorithm is up to the user preferences. They solely depend on what we are more interested in and the heuristics we want to implement. In the reference algorithm, the penalty factors where chosen so that the non allowed states are heavily disregarded. However, that election comes in with a price to pay. If the penalty factors are various orders of magnitude higher than the cost factors, this is $_i; \ _j \gg f_j; c_{ij}$, the norm of the problem Hamiltonian will increase. As we will see later in section 4.8, the norm of the Hamiltonians plays a role on the fidelity of the algorithm. By using the adiabatic theorem (Th. 1) and Eq. 15, we have that, if the di erence in the gap between the initial and the final Hamiltonian increases, derivative term will also increase. This would also increase the annealing time to assure an adiabatic process.

In order to tackle this problem, we propose a new heuristic. The objective is to try to minimize the penalty factors as much as possible, keeping in mind that the penalties still have to make the non allowed states to have higher energies than the solution of the problem. The penalties must be calculated a priori, this is, with no information about the ground state at any point of the evolution. Taking this into account, we propose the following penalty parameters:

i

Any allowed solution must have each client connected to one factory through a supply line. Then, we have to penalize two situations: a client with no connections or with more than one connection. The latter case is penalized by itself, as the cost of having two supply lines at the same time will always be higher than having only one of them. This allows us to focus just on the first situation.

When solving the problem, the algorithm will naturally try to avoid to set up any supply line, as they will always add some cost. Then, to avoid this, we can penalize this situation by finding what

Example 2: Toy problem and its constraint codification

The next step to obtain the problem Hamiltonian is to correctly code the constraints of the problem. For this, we just have to follow Eqs. 60-62 with the correct parameters. In this step, we will leave the penalty factors as a variable, as we will calculate them in the next step.

Then, we can write each of the terms. Note that the terms regarding one constraint act on different subspaces for each element. For example, we can see this in the term from Eq. 60 for different values of i, i.e. for different clients. Then, we can write all these terms separately

$$H_i = {}_i\, 2\, \overset{z}{}_{ij1}\, \overset{z}{}_{ij2} \quad \overset{z}{}_{ij1} \quad \overset{z}{}_{ij2} \;;$$

$$H_1 = {}_1 \quad \overset{z}{}_{i1j1}\, \overset{z}{}_{i1j2} + \overset{z}{}_{k1j1}\, \overset{z}{}_{k1j1} + 2\, \overset{z}{}_{i1j1}\, \overset{z}{}_{k0j1} + \overset{z}{}_{i2j1}\, \overset{z}{}_{k0j1} + 4\, \overset{z}{}_{i1j1}\, \overset{z}{}_{k1j1} + 2\, \overset{z}{}_{i2j1}\, \overset{z}{}_{k1j1} \;;$$

$$H_2 = {}_2 \quad 2\, \overset{z}{}_{i1j2} \quad \overset{z}{}_{i2j2} \quad \overset{z}{}_{k0j2} \quad 2\, \overset{z}{}_{k1j2} + \overset{z}{}_{i1j2}\, \overset{z}{}_{i2j2} + \overset{z}{}_{k1j2}\, \overset{z}{}_{k1j2} + 2\, \overset{z}{}_{i1j2}\, \overset{z}{}_{k0j2} + \overset{z}{}_{i2j2}\, \overset{z}{}_{k0j2} + 4\, \overset{z}{}_{i1j2}\, \overset{z}{}_{k1j2} + 2\, \overset{z}{}_{i2j2}\, \overset{z}{}_{k1j2} \;;$$

$$H_j = {}_j\, 3\, \overset{z}{}_{j} \quad \frac{3}{2}\, \overset{z}{}_{i1j} \quad \frac{3}{2}\, \overset{z}{}_{i2j} \quad 3\, \overset{z}{}_{`0j} \quad 6\, \overset{z}{}_{`1j} \quad \overset{z}{}_{j}\, \overset{z}{}_{i1j} \quad \overset{z}{}_{j}\, \overset{z}{}_{i2j} \quad 2\, \overset{z}{}_{j}\, \overset{z}{}_{`0j} \quad 4\, \overset{z}{}_{`1j} \quad \frac{1}{2}\, \overset{z}{}_{i1j}\, \overset{z}{}_{i2j} + \overset{z}{}_{i1j}\, \overset{z}{}_{`0j} + 2\, \overset{z}{}_{i1j}\, \overset{z}{}_{`1j} + \overset{z}{}_{i2j}\, \overset{z}{}_{`0j} + 2\, \overset{z}{}_{i2j}\, \overset{z}{}_{`1j} + \overset{z}{}_{`0j}\, \overset{z}{}_{`1j} \;!\;:$$

Here we clearly see that the effect of the first and third constraint to not depend on the input parameters of the problem. As these only depends on the problem size, the only distinction between the different terms is the penalty factor. Besides, for the second constraint the terms are clearly different. Each of these terms forms an ATA Hamiltonian on they own. This will be important when we define the topology of the circuit implementing the algorithm.

is the worst case that is still allowed. Then, we can penalize a client with no lines set up with the cost of the most expensive supply line, plus some arbitrary positive constant. As the supply line is not active, a solution with a missing connection does not require to have the corresponding factory built. Since this situation is not corrected in the rest of the constraints, we need to take this into account by adding some construction cost. As we can not know which factory the closest correct solution needs, we can add the cost of all factories. Then, we can set the penalty factor as

$$_i = \max_j c_{ij} + \sum_j f_j + "; \quad " > 0: \tag{63}$$

j

For the second constraint, we have to add a penalty to the solutions asking a factory to dispatch more goods than it produces. In this case it is di cult to think about the closest allowed solution, since changing a supply line from one client interfere with the rest of the configuration. Thus, the solution we propose here is to think of the cost of the worse allowed configuration comprising the rest of the factories. For this, we have to take the cost of building all but the j-th factory and the most expensive supply line arriving to each client. With this naive approximation we are overestimating the cost of the closest allowed configuration, but with the advantage that the computation cost to calculate it is small. The expression for this penalty factor is

$$_j = \sum_{j', j} f_{j'} + \sum_i \max_j c_{ij} + "; \quad " > 0: \tag{64}$$

j

The last constraint takes into account that a factory must be built in order to deliver any goods. In this case, is easy to come up with the penalty factor, as we have to just add a penalty greater than the building cost of the missing factory. Then we can set the penalty factor as

$$_j = f_j + "; \quad " > 0: \tag{65}$$

In all three expressions, we have a variable ", which apparently lacks any physical meaning. Its only purpose is to make the penalty greater than the closest allowed configuration of the state on which it is acting. This means that, in theory, any value grater than zero would work fine. However, in practice, this could lead to a situation in which a pair of allowed and non-allowed states are only separated with a energy di erence of ". If the order of magnitude of " is much smaller than the order of magnitude of the minimum di erence between allowed states, the system could easily jump from an allowed to a non-allowed state. This would be fatal if this situation happens between the ground and the first excited state, as we would need to drastically increase the annealing time (Eq. 15). This problem can be easily avoided by setting this parameter in the same order of magnitude as the supply line and factory costs, " $f_j; c_{ij}$.

Additionally to the penalty factors, we have access to one extra parameter in the Hamiltonian. We can multiply the local ^{x} terms of the initial Hamiltonian (Eq. 18) by a constant, h_0. The chosen heuristic is to reduce the energy of the initial ground state, so that the energies of the initial and final ground states are as close as possible. But there is a problem, we can not know the energy of the final state beforehand. If we wanted to obtain it, we would have to search for it in a 2^N search space, where N is the total number of qubits obtained in the last paragraph of the previous section (sec. 4.2). This would be equivalent to solving the problem with a classical algorithm. To avoid this, we propose the following strategy:

h_0

The first step to make a correct choice for this constant is to estimate the energy of the ground state. This is particularly di cult, the costs and penalty factors interacts with each other via two-qubit coupling terms. The only thing we can predict is that, since the solution for the problem must be allowed, the ground state energy will be at maximum minus the sum of all penalty terms. With this information, and without any a priori information, the choice we made for the factor that multiplies each ^{x} term is

$$h_0 = \frac{1}{N}\left(\sum_i {}_i + \sum_j {}_j + \sum_j {}_j\right): \tag{66}$$

The parameter selection made in this section is neither the only possible one nor the best one. As we have previously mentioned, if we were to chose the optimal parameter set, we would have to solve the eigenvalue problem for the final Hamiltonian. But since the main point of the proposed algorithm is to avoid this calculation, these strategies give an easily calculable set of parameters. The e ciency of the proposed selection of parameters is discussed in the next chapter.

This same heuristic of adjusting the penalty coe cients on a digitized annealing algorithm was also used in Dury and Matteo [45]. However, in that work, they implemented an additional failsafe strategy. If after running a circuit, a non-allowed solution is measured, they search what specific constraint is it violating. Then, they rerun the circuit with the corresponding penalty factor multiplied by 2. This strategy is specially useful in the cases in which the algorithm is set to be run in a real quantum computer. This way, as the circuit is run more times, we obtain a set of optimized parameters. However, as the algorithm of this thesis is not going to be run in a real quantum computer, we leave this strategy aside for the moment.

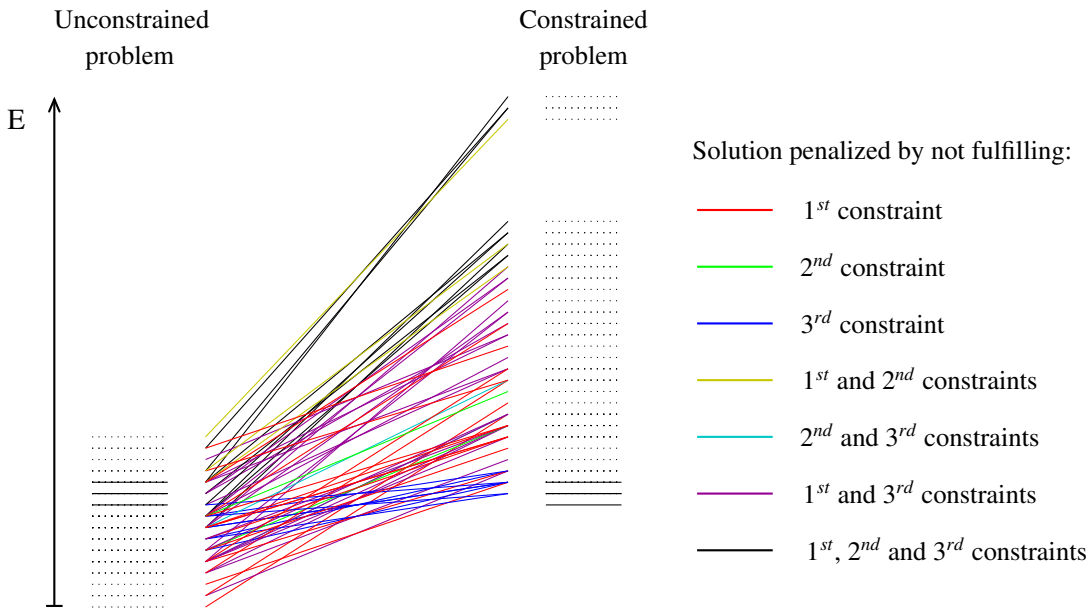
Example 3: Toy problem and the Hamiltonian parameters

On the previous example, we left the penalty factors as parameters. Now we will calculate and discuss them using Eqs. 63-65.

Let us first take the $_i$ penalty factors, which penalize a client with no connections or too many connections. As explained before, we will only focus on penalising the case of a client with no connections. For the first client the most expensive supply line arriving at them has a cost of 4, and the sum of the factory costs is 4. If we make $_1 = 4 + 4 + "$, choosing any of the two supply lines will be a better option than not choosing any. On top of this, the cost of not having the factories built is added to the penalty cost to avoid choosing solutions where no factories are built. Since the order of magnitude of the difference of the cost variables is 1, we set $" = 1$ from now on. We do the same for the second client, obtaining the penalty factors $_1 = 9$ and $_2 = 8$.

For the penalty factors $_j$, we must compensate the clients demanding too much production from a factory. Then, the proposal from Eq. 64 is to search for the worse solution that distribute the client orders to the rest of the factories. In the example we are exploring, this is trivial, as we only have 2 factories. The second factory does not have sufficient goods to make a delivery to both clients at the same time. Then, we have to penalize this situation by adding the cost of building the other factory and the cost of the worse supply lines arriving at each client, $_2 = 3 + 4 + 3 + 1 = 11$. In this case, we see that the first factory allows both clients to ask for supplies, thus we do not have to add the term H_1 in principle. However, in a more realistic problem, it is rare for this to happen. For completeness, we include this term with a penalty factor of $_1 = 1 + 4 + 3 + 1 = 9$.

The last penalty factor is easy to set, as it accounts for demanding goods from a non built factory. Then, to calculate the penalty factors we have to just take the cost of the factory and add $"$, obtaining $_1 = 4$ and $_2 = 2$.



In the plot above it is shown how the penalties increase the energy of the states representing a non allowed configuration. The solid lines here mark the allowed solutions, and the dotted ones the non allowed. The color code for the lines connecting the energy levels indicate what penalties are applied to each non allowed configurations. For the unconstrained problem, only the factory construction and the supply line costs are taken into account. In that case, the trivial optimal solution is to not construct any factory or deliver any goods at all. For the constrained problem, we add the penalties for not fulfilling the constraints. Note that, although some of the non allowed configurations have a final energy lower or equal than some of the allowed ones, the solution for the problem still corresponds to the ground state.

The last parameter is the constant multiplying the initial Hamiltonian, h_0. Before performing any analysis of the evolution of the gap or the fidelity of the problem, we can not optimize this parameter. Thus, we will be following the proposed strategy, summing all the penalties and dividing the result by the number of qubits, obtaining $h_0 = 43{=}14 = 3{:}0714$. With this factor, the Frobenius norm of the initial Hamiltonian is $\|H_0\| = 14h_0 = 42{:}9996$, which is much lower than the norm of the problem Hamiltonian $\|H_P\| = 385$.

4.4 Annealing scheduling and discretization of the evolution

In this section, we discuss the only remaining parameters, the evolution schedule and its discretization. As introduced in section 2.2, we have to chose a time dependent function (t) for Eq. 14. As mentioned before, the boundary conditions for this function are $(0) = 0$ and $(T) = 1$. The task now is to choose the specific function.

We have to take into account that the choice of this function can a ect the consistency of the algorithm. The system is more likely to jump from ground state to the first excited when the gap between them is small. Then, if the (t) function makes the system pass through the point where this gap is the minimum quickly, i.e. diabatically, the probability of staying at the ground state decreases [46]. Then, an objective for the choice is to cross the minimal gap point slowly so that the process is adiabatic. Then, after crossing that point, the system should move away from that point as fast to try to reduce the total annealing time.

There is not a unique manner of finding a (t) function that fulfils these conditions. One way of doing it is by hand, this is, estimating when this minimum gap will occur and trying to adjust the evolution schedule to that assumption. However, this estimation is hard, even more if we have no information about the problem behaviour in advance. If we wanted to do this calculation exactly, we would first need to solve the eigenvalue problem at each time of the evolution. Once again, we have to discard this option as it would mean to solve the problem with a classical algorithm. Recently it has been proposed to use an hybrid quantum-classical algorithm to make this optimization more e ciently [47].

In our case, we have chosen the broadly used linear scheduling, $(t) = t=T$. This way, we reduce the number of free variables, reducing them, in this case, just to the total annealing time T. Furthermore, this function makes the expressions easier to write, as we will see later. Other advantage of this choice is that the derivative term of the Hamiltonian is constant, so calculating the minimum annealing time (Eq. 16) just depends on the ground and excited states at each time. The specific choice of T is discussed in the section 4.8.

As we are using a DAQC/DQC paradigm for the algorithm, we have to discretize the evolution of the system so that it can be simulated evolving it with gates or time independent Hamiltonians. For this, we have the expression from Eq. 45. In that expression, the more time steps dividing the evolution, the more similar to the analog evolution it becomes. The explicit formula for the discretized evolution for the quantum annealing process can be obtained by converting the integral on the exponential into a finite Riemann sum,

$$U_{\text{annealing}} = \mathsf{T} \exp\left(\frac{i}{\sim}\int_0^T H_{\text{tot}}(t)dt\right) \qquad U_{\text{discrete}} = \mathsf{T}\prod_{k=1}^{n_T} \exp\left(\frac{i\ t}{\sim} H_{\text{tot}}(k\ t)\right); \tag{67}$$

where n_T is the number of steps of equal length $t = T=n_T$ in which we have divided the total evolution time, and the time ordering operator on the right-hand side just indicates that the steps must be applied in order, being the term with $k = 1$ on the rightmost position. Note that the discrete evolution starts on the step $k = 1$ instead of $k = 0$. Since $H_{\text{tot}}(0) = H_0$, the initial state is an eigenstate of the initial Hamiltonian, and the evolution under H_0 is equal for all qubits. Then, applying the first step would only introduce a global phase of $e^{\ i\ t h_0=\sim}$ which can be ignored.

Although we have an expression to calculate an upper limit for the Trotterization error, it is hard to directly relate it to the probability of the annealing process to fail. The correct number of steps to have a reasonable probability of measuring the correct state at the end of the algorithm heavily depends on the problem. For example, in Barends et al. [48], they just divide the evolution into 5 steps to obtain a 4 qubit GHZ state. Other example is given by Mbeng et al. [49], where they use $2N$, with N the total number of qubits, to perform a digitized QA process for the Ising antiferromagnetic chain. As it is di cult to predict what should be the correct number of steps, we have performed various simulations with varying number of steps. The results of these simulations are shown later in section 5.2.

4.5 Annealing Hamiltonian

Now that all parameters and terms have been discussed, we can write the full Hamiltonian for the annealing process. For this, we just add the terms from equations 59 to 62 and the local terms of the initial Hamiltonian with the corresponding factors from equations 63 to 66. The Hamiltonian is then

$$H_{\text{tot}}(t) = \left(1 - \frac{t}{T}\right) H_0 + \frac{t}{T} H_P; \tag{68}$$

with the initial Hamiltonian

$$H_0 = h_0 \sum_j \hat{x}_j + \sum_j \hat{x}_j + \sum_{ij} \hat{x}_{ij} + \sum_{kj} \hat{x}_{kj}; \tag{69}$$

and the problem Hamiltonian

$$\begin{aligned} H_P =& H_{\text{cost}} + H_{\text{cost}} + H_{\text{cost}} + H_{\text{cost}} = \sum_j 2^{\grave{}_j - 1}(2^L - 1)\, \hat{z}_j + \frac{I - \sum_j (2^L - 1)}{2} \prod_j \hat{z}_j \\ &+ \sum_i (3 - 2J) \left(\frac{d_i - \sum_j}{2} \sum_i d_i - 2v_j + 2^{K_j - 1} \right) \sum_j (2^L - 1) \sum_{ij} \hat{z}_{ij} \\ &2^{k-1} \sum_j \left(\sum_i d_i - 2v_j + 2^{K_j - 1} \right) \sum_{kj} \hat{z}_{kj} + 2 \sum_i \sum_{ij1} \hat{z}_{ij2} + \frac{\sum_j d_{i1} d_{i2} + \sum_j}{2} \sum_{i1j} \hat{z}_{i2j} \\ &+ 2^{k1+k2-1} \sum_j \hat{z}_{k1j} \hat{z}_{k2j} + 2^{k-1} \sum_j d_i \sum_{kj} \hat{z}_{ij} - \frac{\sum_j I}{2} \sum_j \hat{z}_{ij} - \frac{I - \sum_j}{2} \sum_j \hat{z}_{\grave{}_j} \\ &+ 2^{\grave{}_1 + \grave{}_2 - 1} \sum_j \hat{z}_{\grave{}_{1j}} \hat{z}_{\grave{}_{2j}} + 2^{\grave{}_j - 1} \sum_j \hat{z}_{ij} \hat{z}_{\grave{}_j}. \end{aligned} \tag{70}$$

Now that we have the explicit form of the Hamiltonian, we have to find an expression for its time evolution that can be implemented by using the DQC or DAQC paradigms. For this, we will apply the Trotter formula to separate each of the steps of the discretized evolution (Eq. 67) into terms that can be simulated. For this, we could take the Trotter expansion at first order (Eq. 45). However, it is more sensible to take the symmetric version from Eq. 40,

$$\begin{aligned} U_{\text{annealing}} &= \mathcal{T} \exp\left[\frac{i}{\sim} \int_0^T H_{\text{tot}}(t) dt \right] \qquad U_{\text{discrete}} = \mathcal{T} \prod_{k=1}^{n\grave{}-1} \exp\left[\frac{i\ t}{\sim} H_{\text{tot}}(k\ t) \right] \\ U_{\text{discrete+Trotter}} &= \mathcal{T} \prod_{k=1}^{n\grave{}-1} \exp\left[\frac{i\ t(1 - (k\ t))}{2\sim} H_0 \right] \exp\left[\frac{i\ t\ (k\ t)}{\sim} H_P \right] \exp\left[\frac{i\ t(1 - (k\ t))}{2\sim} H_0 \right]. \end{aligned} \tag{71}$$

Introducing the explicit formula for (t) we obtain

$$\begin{aligned} U_{\text{d+T}} &= \mathcal{T} \prod_{k=1}^{n\grave{}-1} \exp\left[\frac{i\ t(1 - \frac{k\ t}{T})}{2\sim} H_0 \right] \exp\left[\frac{i\ t\frac{k\ t}{T}}{\sim} H_P \right] \exp\left[\frac{i\ t(1 - \frac{k\ t}{T})}{2\sim} H_0 \right] \\ &= \mathcal{T} \prod_{k=1}^{n\grave{}-1} \exp\left[\frac{i\ t(nT - k)}{2nT\sim} H_0 \right] \exp\left[\frac{i\ tk}{nT\sim} H_P \right] \exp\left[\frac{i\ t(nT - k)}{2nT\sim} H_0 \right]. \end{aligned} \tag{72}$$

We can reduce the number of evolution steps with H_0 by writing them explicitly and merging the adjacent terms using the BHC formula,

$$\begin{aligned} U_{\text{d+T}} =& e^{\frac{i\ t}{2nT\sim} H_0} e^{\frac{i\ t(nT\ 1)}{nT\sim} H_P} e^{\frac{i\ t}{2nT\sim} H_0} \quad e^{\frac{i\ t2}{2nT\sim} H_0} e^{\frac{i\ t(nT\ 2)}{nT\sim} H_P} e^{\frac{i\ t2}{2nT\sim} H_0} \\ & e^{\frac{i\ t(nT\ (k+1))}{2nT\sim} H_0} e^{\frac{i\ t(k+1)}{nT\sim} H_P} e^{\frac{i\ t(nT\ (k+1))}{2nT\sim} H_0} \quad e^{\frac{i\ t(nT\ k)}{2nT\sim} H_0} e^{\frac{i\ t(nT\ k)}{nT\sim} H_P} e^{\frac{i\ t(nT\ k)}{2nT\sim} H_0} \quad e^{\frac{i\ t(nT\ 1)}{2nT\sim} H_0} e^{\frac{i\ t}{nT\sim} H_P} e^{\frac{i\ t(nT\ 1)}{2nT\sim} H_0} \\ =& e^{\frac{i\ t}{2nT\sim} H_0} e^{\frac{i\ t(nT\ 1)}{nT\sim} H_P} e^{\frac{i\ t(\ 3)}{2nT\sim} H_0} e^{\frac{i\ t(nT\ 2)}{nT\sim} H_P} \quad e^{\frac{i\ t(k+1)}{nT\sim} H_P} e^{\frac{i\ t(2nT\ (2k+1))}{2nT\sim} H_0} e^{\frac{i\ tk}{nT\sim} H_P} \quad e^{\frac{i\ t(2nT\ 3)}{2nT\sim} H_0} e^{\frac{i\ t}{nT\sim} H_P} \\ =& e^{\frac{i\ t}{2nT\sim} H_0} e^{\frac{i\ t(nT\ 1)}{nT\sim} H_P} \mathcal{T} \prod_{k=1}^{n\grave{}-2} e^{\frac{i\ t(2nT\ (2k+1))}{2nT\sim} H_0} e^{\frac{i\ tk}{nT\sim} H_P}; \end{aligned} \tag{73}$$

where we have taken out the first evolution with H_0 since, again, it only introduces a global phase to the initial state.

4.6 Initial state preparation and measurement of the final state

Roughly speaking, a quantum algorithm has three main sections: initial state preparation, state evolution and the measurement of the final state. We have already talked in depth about the steps for the evolution of the state. In this section, we will present the two parts of the algorithm that are left.

As introduced in section 2.2, the system must be initialized in the ground state of the initial Hamiltonian. For this reason, we saw that the initial Hamiltonian is chosen so that its ground state is easily preparable. In this case, since we have chosen to use a a initial Hamiltonian with only local ^{x} terms, the ground state fulfils this requirement. The ground state of a ^{x} Hamiltonian is $jgi = (j0i \quad j1i)=\sqrt{2}$, but SCC are usually reseted to the state $jg_{SCC}i = j0i^{N}$. Then, we can prepare the initial state by applying 2 SQG to each qubit. In this case, we have to apply a Hadamard gate followed by a Z gate, $j_{in}i = (ZH)^{N} j0i^{N}$.

When the operations to evolve the system ends, we are left with a state containing the information about the solution to our problem. In order to extract this information we have to measure the quantum state. There are various ways of doing this, for instance, quantum tomography, that extract the full information about the quantum state [50], or quantum non-demolition measurements, which preserves the state in the measured eigenstate [51]. Fortunately, our algorithm just requires a simple measurement of the state in the canonical basis.

We have used a number of ancilla qubits to introduce information about the problem to the system. However, we are only interested in the information about the information about the logistic network configuration. This information is stored in the qubits labeled with j for the built factories and ij for the active supply lines, as seen in section 4.1. Then, we only have to measure each of these qubits on the projection of the z axis. The ancilla qubits can be discarded without loosing any relevant information. Once we have obtained this information, we can directly convert it to the classical representation of the problem solution in terms of the binary vector x and matrix y, described in section 3.1.

4.7 Proposed SCC topology

If we were to simulate the evolution described in section 4.5, we could use any SCC topology. In particular, we could simulate it in a circuit with the connections of a NN Hamiltonian, as shown in section 2.3.1. However, this scheme would require a vast amount of Swap gates to be implemented. In order to try to reduce the number of swap operations while maintaining the circuit topology su ciently simple, we make the following proposal.

In the introduction to DQC, we saw that the evolution of a local Hamiltonian can be implemented in a circuit by applying just one single qubit rotation gate. In this case, as H_0 only has local terms, its evolution can be implemented with one SQG per qubit. This leaves us only with the problem of simulating the evolution of H_P.

As discussed in section 2.3.1, in order to simulate an ATA Hamiltonian acting on N qubits, we need to simulate $dN=2e$ NN Hamiltonians, which in turn have to be simulated using $N \quad 1$ analog blocks. This means that, in total, we need at most $dN=2e(N \quad 1)$ analog blocks to simulate the problem Hamiltonian of Eq. 70. We can improve the number of blocks needed from this naive solution.

If we analyse the two-qubit interactions on the problem Hamiltonian, we can see that the interacting terms only act on certain qubit subspaces, which can be divided into three groups: $f`j; j; ijg$ with the same j, $fij; kjg$ with the same j and $fijg$ with the same i. Each of the subspaces of the same kind is disconnected from the rest, allowing for a parallel realization. As an example for this, lets take the subspace $fij; kjg$.

For di erent j, the supports do not superpose. Then, we can treat each one as an independent ATA Hamiltonian, allowing us to simulate their evolution in parallel.

However, note that $\overset{z}{i1\,j}$ $\overset{z}{i2\,j}$ and $\overset{z}{i\,j}$ appear in two di erent ATA terms. In order to keep the ATA structure on both terms, we can divide them into two, assigning one to each of the ATA Hamiltonians. As all the interacting terms are on the z axis, every term commute with each other. This allows us to generate the ATA Hamiltonians in arbitrary order without changing the result.

Then, every ATA Hamiltonian can be generated with NN Hamiltonians. This leads to the circuit topology depicted in Fig. 10. The proposed realistic topology fits well with the current state-of-art. The topology of the couplings can be represented by a plane graph, thus the circuit could be integrated into the usual silicon wafers. Furthermore, the maximum number of couplings for a single qubit is four, a number of couplings that has been achieved years ago. An example of this kind of technology can be found in IBM's *ibmq_5_yorktown* computer and in D-wave's annealers.

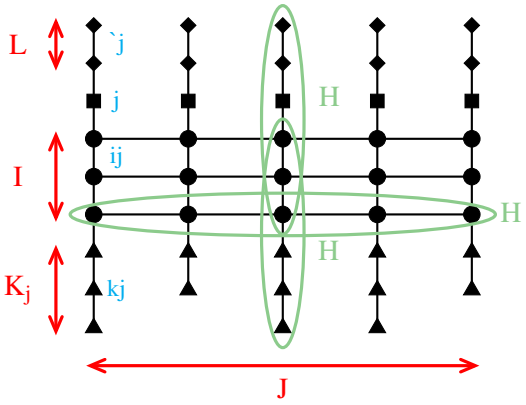


Fig. 10: Proposed topology for implementing the digitized QA algorithm for the LND problem. The qubit labels are marked in blue, which corresponds to the notation used in this chapter. The sizes of the qubit sets are shown in red, and they depend on the problem size or its parameters (sec. 4.2). Note that the second and fth columns have one qubit less in the subspace of f$k\,j$g, remarking that their size might vary depending on the input parameters of the problem. The di erent terms from the problem Hamiltonian are marked in green, where each bubble highlights each of the ATA Hamiltonians separately simulated (sec. 4.5). The ideal circuit topology for this algorithm has the qubit connectivity given by ATA Hamiltonians, which corresponds with a non-planar graph.

After setting the evolution of the ATA Hamiltonians applied in parallel, the total time required for simulating of this process would be the time for the slowest term. In a implementation on a real quantum computer, this strategy allows for a simulation of more time steps without reaching the computation time limit, i.e. the decoherence time.

Additionaly, we can not forget that the time to apply a quantum gate in a SCC platform is not zero. We can do a rough estimation of the total number of required gates in order to give an approximate time for the algorithm. For an ATA Hamiltonian of size N, we need to apply dN=2e $F(k;N)$ blocks. Each of them can be implemented applying two qubit gates N 1 times, as some of the Swap gates can be applied in parallel. Then, for each NN Hamiltonian, we need to apply the $X_j^{f_j(k)}$ blocks N times. This sums up to dN=2e two-qubit gates and dN=2eN single qubit gates. We have to repeat this for the three types of ATA Hamiltonians in which we have divided the interactions, multiplying this by the number of steps for the evolution discretization.

For IBM's quantum computers, the time to apply a two-qubit gate is 100 1000 ns depending on the circuit type, while one qubit gates are applied in 10 100 ns. As we pointed out before (sec. 2.1), the best decoherence time we can achieve with IBM computers is 100 s. This is a huge drawback

for the implementation in a real quantum computer, as the running time of the algorithm must be much shorter than the decoherence time. With the currently available technology, we are consequently limited to solve small problems comprising a few qubits.

Taking these considerations into account, we have to choose an annealing time T. The main consideration here, apart from reaching the desired fidelity, is to choose T so that the running time of the algorithm is su ciently small to fit into the coherence time of the physical device. As shown in Fig. 11, the total time for the evolution increases linearly with the annealing time and the number of steps n_T. The time will also depend on the strength of qubit couplings. Indeed, the stronger the coupling, the shorter the time for the analog blocks to be applied.

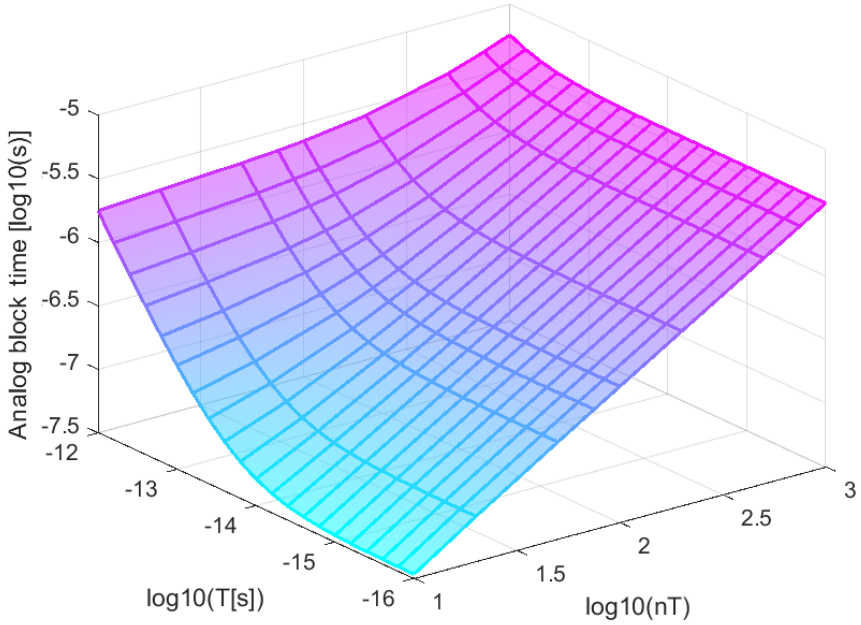


Fig. 11: Logarithmic plot of the total time for the evolution of the analog blocks vs the annealing time T and the number of steps n_T. The values have been taken for a SCC of 14 qubits solving the toy problem (Ex. 1). For the SCC we have used the realistic topology proposed in Fig. 10 with random couplings of strength (3:4 1:7) $10^{-5}eV$ = (8:2 4:1)GHz (value of the same order of magnitude as the couplings from the device described in Grajcar et al. [11]). We have taken into account the time of the analog blocks for the evolution of the target NN Hamiltonians and the time to implement the Swap gates from Eq. 20. From that equation, we can calculate the depth of the circuit implementing complete set of $F(k;n)$ blocks for an ATA Hamiltonian, which upper bound is $O(n) = 2(n + dn{=}2e \quad 2)$. The results shows that the analog time increases linearly with both n_T and T. For small annealing times, the term contributing more to the total time for the analog blocks is the time for implementing the Swap gates. We have not taken into account the time to apply any SQG, which in a realistic scenario will add a non negligible time to the time to complete the algorithm.

After this discussion, we have to discard the implementation in a quantum computer with state-of-art technology. Then, we can to aim to a more ideal case, in which we can get over some of the limitations complicating the implementation. For this, we propose a circuit topology that perfectly matches the Hamiltonian from Eq. 68. The proposed ideal topology is sketched in Fig. 10. If we were to run the circuit with the ideal topology using the DAQC framework, we would only need one analog block to simulate the evolution of H_P at each step, without the need of applying any Swap operation. The ideal topology is particularly useful in the DQC framework. As seen in the introduction (sec. 2.1), we just need two two-qubit gates to simulate the evolution of an interacting two qubit term. Then, we can easily

obtain the number of two-qubit gates of the circuit by just counting the number of interactions on H_P and multiplying it by $2n_T$. For this, we will count the number of interactions of each ATA Hamiltonian (the same as the edges in a complete graph), and then subtract the repeated interactions,

$$\#\text{two-qubit gates} = 2n_T \left(J\frac{(L+I+1)(L+I)}{2} + \sum_{j=1}^{} \frac{(I+K_j)(I+K_j \quad 1)}{2} + I\frac{J(J \quad 1)}{2} \quad J\frac{I(I \quad 1)}{2} \right): \tag{74}$$

However, the number of gates is not equal to the depth of the circuit, which can be defined as the total number of resources needed to complete the circuit. As we have seen in section 2.1, the time for applying a SQG is shorter than the time to apply a two-qubit gate. Although it is just an approximation at all, we can assume that the only resources we need to take into account are the two-qubit gates. Then, we can define the depth of our circuit as the number of time steps at which we have to apply a two-qubit gate.

If we assume that we are using the circuit with the ideal topology, we can apply two two-qubit gates given that they act on 4 di erent qubits. The depth of a circuit implementing the evolution of an N qubit ATA Hamiltonian on a circuit with the topology of a complete graph is $N \quad 1$ for N even and N for N odd. We can use a similar result for an arbitrary Hamiltonian, obtaining that the depth of the circuit is given by the maximum number of connections a qubit has. For our particular problem, the qubits with more connections are the ones labeled with ij, with a total number of connections of $J+I+L+\max_j K_j$. Taking this as the depth of the circuit simulating one time step, we see that the depth of the circuit increases linearly with the number of factories and clients, and logarithmically with the number of factories and the production of the factories. Lets make the following reasonable assumption: for a large problem, the number of clients is much larger than the number of factories, and the production of the factories is on the scale of the number of clients. In this limit, we can say that the depth of the problem grows linearly with the number of clients and the number of discretization steps,

$$O_{\text{depth}}(I;J;v_j;nT) = 2n_T \quad J+I+L+\max_j K_j \quad I \quad J;K_j \quad 1 \quad O(I;n_T) = 2n_T I: \tag{75}$$

Lets also recap here the result from section 4.2 about the number of qubits to implement the algorithm. We see that, in this same limit, the number of qubits needed increases linearly with the number of factories and clients,

$$O_{\text{qubits}}(I;J;v_j) = J(I+L+1) + \sum_j K_j \quad I \quad J;K_j \quad 1 \quad O_{\text{qubits}}(I;J) = IJ: \tag{76}$$

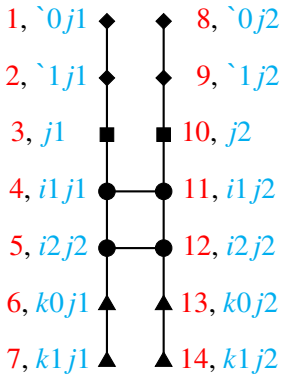
4.8 Fidelity and errors

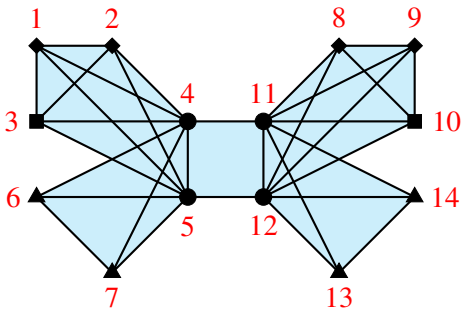
As in every other quantum algorithm, we are interested in the rate of success of the proposed algorithm, i.e. we want to know what is the probability of measuring the desired result after running the quantum circuit. This is given by the fidelity F, although some times is useful to give the failure rate, the infidelity $(1 \quad F)$.

The fidelity of an algorithm is related to all the errors introduced in the circuit. For the case of our algorithm, we have three main error sources: the error on the annealing process, the discretization error, and the Trotterization error. The analysis of this section holds for the case of a decoherence free quantum computer with perfect measurements. We will briefly address how to possibly deal with this kind of errors in the conclusion of the thesis.

Example 4: Circuit topology for the toy problem

We have now all all the tools to build the Hamiltonian for the toy problem, so lets provide the proper circuit to simulate it. In this example, we give two solutions to the problem. The naive one is the ideal circuit, which has the same couplings as the interacting terms of the problem Hamiltonian. Even though we are working with the smallest non-trivial problem, the circuit topology required is not planar. The most practical topology for the problem consist on a circuit with a square lattice for the couplings. Nevertheless, we have to pay a price for using this more affordable topology, we have to simulate each of the ATA Hamiltonian terms that arises from the Hamiltonian with NN Hamiltonians as a source, as explained in section 2.3.1.





The figures depicted here represent both circuit topologies, namely, the realistic topology on the left and the ideal topology on the right. For clarity, we have maintained the same notation as the one used in Fig. 10 and along the examples.

Annealing error: The error on the annealing process is the unavoidable error that comes from the non-zero probability of exciting the ground state. As introduced in section 2.2, this depends on the minimum gap between the ground and the first excited states along the evolution, and the rate at which these states approach each other. The fidelity of the annealing process is hard to calculate. However, as shown in Eq. 15, the process will arrive at the final ground state given su cient time.

Discretization error: The discretization error is the error introduced by approximating the time ordered integral of the annealing process to a stepwise evolution with time independent Hamiltonians (Eq. 45). This can be calculated as the di erence between the continuous evolution and the discrete evolution, with a expression similar to the one used to calculate the error introduced when approximating an integral by its Riemann sum. The fidelity of the discretization on the k-th step is

$$F_{1\text{step}} = \langle k | \exp\left[\frac{i}{\hbar}\int_{k\,t}^{(k+1)\,t} H_{\text{tot}}(t)dt\right] \exp\left[-\frac{i\,t}{\hbar} H_{\text{tot}}(k\,t)\right] | k \rangle^2 ; \tag{77}$$

where $|k\rangle$ is the ground state of the annealing Hamiltonian at the time $t = k\,t$. Now, we can give

an easier manner to calculate formula if we use the BCH formula at first order,

$$\begin{aligned} F^{(1)}_{\text{1step}} &= \langle \psi_k | \exp\left[\frac{i}{\sim} \int_{k\ t}^{(k+1)\ t} H_{\text{tot}}(t)dt \quad \frac{i\ t}{\sim} H_{\text{tot}}(k\ \ t) \right] | \psi_k \rangle^2 \\ &= \langle \psi_k | \exp\left[\frac{i\ t}{\sim} \frac{H((k+1)\ \ t) \quad H(k\ \ t)}{2} \right] | \psi_k \rangle^2 \\ &= \langle \psi_k | \exp\left[\frac{i\ t}{\sim} \frac{(n_T \quad k \quad 1)H_0 + (k+1)H_P \quad (n_T \quad k)H_0 \quad kH_P}{2n_T} \right] | \psi_k \rangle^2 \\ &= \langle \psi_k | \exp\left[\frac{i\ t}{\sim} \frac{H_P \quad H_0}{2n_T} \right] | \psi_k \rangle^2 : \end{aligned} \tag{78}$$

Since the fidelity is independent on the step number, we can straightforwardly extend this result to the fidelity of the discretization of the whole integral

$$F_{\text{discrete}} = \langle \psi_{in} | U^{\dagger}_{\text{annealing}} U_{\text{discrete}} | \psi_{in} \rangle^2 \quad F^{(1)}_{\text{discrete}} = \langle \psi_{in} | \exp\left[\frac{i\ t}{\sim} \frac{H_P \quad H_0}{2} \right] | \psi_{in} \rangle^2 ; \tag{79}$$

where $U_{\text{annealing}}$ and U_{discrete} are taken from Eq. 71. As expected, the discretization error will get smaller with $\ t$. However, we got this result after approximating the expression at the first order of the BCH. To see when this approximation holds, we can repeat the same process but employing the second term of the BCH formula

$$F^{(2)}_{\text{discrete}} = \langle \psi_{in} | \exp\left[\frac{i\ t}{\sim} \frac{H_P \quad H_0}{2} \quad \frac{t^2}{4\sim^2 n_T} [H_0; H_P] \right] | \psi_{in} \rangle^2 : \tag{80}$$

If we take here the norm of the terms on the exponential, we can assure that the fidelity expression for the discretization from Eq. 79 will hold if $\|H_P \quad H_P\| \quad t\|[H_0; H_P]\| = 2\sim n_T$.

Trotterization error: This error comes from using Eq. 40 to split the evolution of each time dependent steps. In this case, since we already got the expressions for the discrete evolution and its Trotter expansion (Eq. 71), we can write the fidelity as

$$F_{\text{Trotter}} = \langle \psi_{in} | U^{\dagger}_{\text{discrete}} U_{\text{d+T}} | \psi_{in} \rangle^2 ; \tag{81}$$

where $U_{\text{d+T}}$ is taken from Eq. 71. As we have seen in section 2.5, the error of the first order Trotterization will grow with $(\ t=\sim)^3 \|[H_0; H_P]\|$. This gives us an estimation to predict whether the Trotterization error can be neglected or not.

Digitization error: In the previous paragraphs we have addressed the di erent sources of errors separately. This way, we could apply error propagation theory to give an upper bound as the sum of the errors. However, in the cases of discretization and Trotterization errors, we can get more information addressing both simultaneously. Both errors could be cancelled, or at least mutually mitigated. Indeed, discretization and Trotterization errors are the error made when we approximate the continuous annealing process for the DQC or DAQC paradigms, i.e. the error introduced when digitizing the annealing process. Then, the fidelity after simultaneously applying these two approximations is

$$F_{\text{d+T}} = \langle \psi_{in} | U^{\dagger}_{\text{annealing}} U_{\text{d+T}} | \psi_{in} \rangle^2 = \langle \psi_{in} | U^{\dagger}_{\text{annealing}} U_{\text{discrete}} U^{\dagger}_{\text{discrete}} U_{\text{d+T}} | \psi_{in} \rangle^2 : \tag{82}$$

If we insert here the result from Eq. 79, we obtain a explicit formula that can be calculated numerically

$$F_{\text{d+T}} \quad F^{(1)}_{\text{d+T}} = \langle \psi_{in} | \exp\left[\frac{i\ t}{\sim} \frac{H_P \quad H_0}{2} \right] U^{\dagger}_{\text{discrete}} U_{\text{d+T}} | \psi_{in} \rangle^2 : \tag{83}$$

4.9 Follow-up problem: LND with loose constraints

In a realistic scenario, a solution that has a good cost but is not allowed by a small distance can be preferable to any other allowed solution with a worse cost. To solve real life logistic problems, the initial plans can be reconsidered if that gives some kind of advantage. As logistics has a large number of parameters one has to take into account, the simplistic LND model used in this thesis fails to consider relevant factors. However, some of them, as the time of delivery or cuts in the supply lines, are di cult to be correctly quantified. To show the adaptability of our algorithm, we propose to add a new set of more realistic constraints. For that, we can overwrite some of the constraints to take into account that some of the requirements may be relaxed.

For example, we can allow a client to take goods from more that one factory. This is usually the case for logistics, where redundancy helps to overcome delays or other type of problems. However, we can think that this comes with a price to pay. For each supplier a client has, we can add an extra unpacking cost u. The new cost function will be then

$$F_u(x;y) = \sum_{i,j} u_i y_{ij}. \tag{84}$$

But now, we would need to address the fact that the amount of goods incoming from a supply line does not strictly depend on the demand of the client. To correctly distribute the production, we need to specify the amount of goods transported in each supply line. For this, we can introduce a new variable that tracks the goods transported in each supply line. The most e cient way of code it into the system is by using a binary codification. Now, each binary variable y_{ij} unfolds into $M_i = \lceil \log_2(d_i + 1) \rceil$ binary variables y_{ijm}, $m = 0; 1; \dots; M_i - 1$. We can further assume that the cost of setting the supply line is constant, regardless of the amount of goods transported. However, to make the problem more realistic, we can add a cost that depends on the weight of the cargo (i.e. the amount of goods transported), w_{ij}. To keep the Hamiltonian as simple as possible, we leave the variable y_{ij} as a binary variable that codes whether or not a supply line is active. Then, the total transport cost will be given by

$$F_t(x;y) = \sum_{i,j} c_{ij} y_{ij} + \sum_{i,j,m} w_{ij} 2^m y_{ijm}. \tag{85}$$

To connect both set of variables, we need to add a a new constraint, regarding that a if any amount of goods is transported on a supply line, it must be active. By using the same strategy as for the original third constraint (Eq. 65), the states that doesn't fulfill this condition are penalized,

$$y_{ijm} \quad y_{ij}; \ \forall i; \ \forall j; \ \forall m. \tag{86}$$

We can also accept that a factory can produce more than it originally produces if we make a greater investment on it. We can justify this by thinking that a factory can invest on more workers or machinery to increase its production. This option could be more convenient than setting a full new factory to supply a little amount of extra goods. We can set a new constraint that sets the cost g_j of each extra unit of production as a quadratic function. If we take into account the previous modification, the new constraint will be specified as a new term in the cost function

$$F_g(x;y) = \sum_j \begin{cases} 0 & \text{if } \sum_i d_i y_{ij} \quad v_j; \\ g_j \cdot \left(\sum_{i,m} d_i 2^m y_{ijm} \quad v_j \right)^2 & \text{otherwise:} \end{cases} \tag{87}$$

If we want to address both modifications at the same time we have to rework the Hamiltonian. Now, the only initial constraint left is the third one, while the rest have been absorbed as part of the cost. From this point, we proceed as before. We rewrite the binary variables as terms in a Hamiltonian following the same steps as in sections 4.1 and 4.2:

The original cost function/Hamiltonian H_{cost} from Eq. 59 is left unchanged, as well as the penalty term H from Eq. 62.

The Hamiltonian for term F_p can be written simply as a local term applying the transformation $\hat{q} = (\mathbb{1} \quad ^z)/2$,

$$H_u = \frac{u_i}{2} \; {}^z_{ij}. \tag{88}$$

The transport cost now consist on two terms: the supply line set cost and the weight fare. Using the expression from Eq. 85, the Hamiltonian term is

$$H_t = c_{ij}\hat{q}_{ij} + w_{ij}2^m\hat{q}_{ijm} = \frac{c_{ij}}{2} \; {}^z_{ij} \quad w_{ij}2^m \; {}^1 \; {}^z_{ijm}. \tag{89}$$

To write term for the constraint regarding the transport (Eq. 86) we can follow the same idea as we did with the third constraint of the original problem (Eq. 62). In order to rewrite the inequality as an equality, we need to add a slack variable q_{ijr}, which can be e ciently coded as a binary number. In this case, we need to introduce $V_i = \lceil \log_2(M_i + 1) \rceil$ ancilla qubits per each y_{ij} term, labeled with $v = 0, \ldots, V_i \quad 1$. This term will be

$$\begin{aligned}
&\left(\sum_{ij} \left(V_{ij}y_{ij} \quad \sum_m y_{ijm} \quad \sum_v 2^v q_{ijv} \right)^2 \right) \quad H = \sum_{ij} \left(V_{ij}\hat{q}_{ij} \quad \hat{q}_{ijm} \quad 2^v\hat{q}_{ijv} \right)^2 \\
&= \sum_{ij} \; V_{ij}\hat{q}_{ij}^2 \quad 2V_{ij}\hat{q}_{ij}\hat{q}_{ijm} \quad V_{ij}2^{v+1}\hat{q}_{ij}\hat{q}_{ijv} + \hat{q}_{ijm}^2 + 2\hat{q}_{ijm1}\hat{q}_{ijm2} + 2^{v+1}\hat{q}_{ijm}\hat{q}_{ijv} \\
&\quad +2^{2v}\hat{q}_{ijv}^2 + 2^{v1+v2+1}\hat{q}_{ijv1}\hat{q}_{ijv2} \\
&= \frac{ij}{2} \left(V_{ij}^2 \; {}^z_{ij} \quad V_{ij} \; {}^z_{ij} \; {}^z_{ijm} \quad {}^z_{ij} \quad {}^z_{ijm} \quad V_{ij}2^v \; {}^z_{ij} \; {}^z_{ijv} \quad {}^z_{ij} \quad {}^z_{ijv} \quad {}^z_{ijm} \right. \\
&\quad + \; {}^z_{ijm1} \; {}^z_{ijm2} \quad {}^z_{ijm1} \quad {}^z_{ijm2} + 2^v \; {}^z_{ijm} \; {}^z_{ijv} \quad {}^z_{ijm} \quad {}^z_{ijv} \\
&\quad \left. 2^{2v} \; {}^z_{ijv} + 2^{v1+v2} \quad {}^z_{ijv1} \; {}^z_{ijv2} \quad {}^z_{ijv1} \quad {}^z_{ijv2} \right) \\
&= \frac{ij}{2} \left(V_{ij} \quad V_{ij} + M_i + 2^{V_i} \quad 1 \quad {}^z_{ij} \quad V_{ij} + M_i + 2^{V_i} \quad 1 \quad {}^z_{ijm} \quad 2^v \quad V_{ij} + M_i + 2^{V_i} \quad 1 \quad {}^z_{ijv} \right. \\
&\quad \left. V_{ij} \; {}^z_{ij} \; {}^z_{ijm} \quad V_{ij}2^v \; {}^z_{ij} \; {}^z_{ijv} + \; {}^z_{ijm1} \; {}^z_{ijm2} + 2^v \; {}^z_{ijm} \; {}^z_{ijv} + 2^{v1+v2} \; {}^z_{ijm1} \; {}^z_{ijm2} \right);
\end{aligned} \tag{90}$$

where $m1 < m2$ and $v1 < v2$. The minimum penalty we have to add to the energy in this case is easy to deduce, as we only have to add the cost of setting up the supply line plus some constant

$$_{ij} = c_{ij} + \; ; \quad > 0. \tag{91}$$

For the term F_g we need to introduce again some new slack variables q_{kj}. In this case, the inequality comes from the conditional extra production cost. Fortunately, these slack variables are exactly

of the same form as the ones introduced for Eq. 61. Thus, we can write

$$
\begin{aligned}
F_g &= \sum_j g_j \Big(\sum_{i,m} d_i 2^m y_{ijm} + \sum_k 2^k q_{kj} \quad v_j \Big)^2 \quad ! \quad H_g = g_j \ d_i 2^m \hat{q}_{ijm} + 2^k \hat{q}_{kj} \quad v_j \ ^2 \\
&= g_j \ d_i^2 2^{2m} \hat{q}_{ijm}^2 + 2 d_{i1} d_{i2} 2^{2m} \hat{q}_{i1jm} \hat{q}_{i2jm} + 2 d_i^2 2^{m1+m2} \hat{q}_{ijm1} \hat{q}_{ijm2} + d_{i1} d_{i2} 2^{m1+m2} \hat{q}_{i1jm1} \hat{q}_{i2jm2} \\
&\quad + 2 d_i 2^{m+k} \hat{q}_{ijm} \hat{q}_{kj} \quad d_i v_j 2^m \hat{q}_{ijm} + 2^{2k} \hat{q}_{kj}^2 + 2^{k1+k_2} \hat{q}_{k1j} \hat{q}_{k2j} \quad 2^k v_j \hat{q}_{kj} \\
&= g_j \Big[d_i^2 2^{2m \ 1} \ z_{ijm} + d_{i1} d_{i2} 2^{2m \ 1} \ z_{i1jm} \ z_{i2jm} \quad z_{i1jm} \quad z_{i2jm} + d_i v_j 2^{m \ 1} \ z_{ijm} \quad 2^{2k \ 1} \ z_{kj} \\
&\quad + d_i^2 2^{m1+m2 \ 1} \ z_{ijm1} \ z_{ijm2} \quad z_{ijm1} \quad z_{ijm2} + d_{i1} d_{i2} 2^{m1+m2 \ 2} \ z_{i1jm1} \ z_{i2jm2} \quad z_{i1jm1} \quad z_{i2jm2} \\
&\quad + d_i 2^{m+k \ 1} \ z_{ijm} \ z_{kj} \quad z_{ijm} \quad z_{kj} + 2^{k1+k2 \ 1} \ z_{k1j} \ z_{k2j} \quad z_{k1j} \quad z_{k2j} + v_j 2^{k \ 1} \ z_{kj} \Big] \\
&= g_j \Big[d_i 2^{m \ 1} \Big(2^m \quad 2^{M_i} + 1 \quad \sum_{i',i} d_{i'} + 2^{K_j} \quad v_j \quad 1 \Big) z_{ijm} + d_i^2 2^{m1+m2 \ 1} \ z_{ijm1} \quad z_{ijm2} \\
&\quad 2^{k \ 1} \Big(2^{M_i} \quad 1 \quad \sum_i d_i + 2^{K_j} \quad v_j \quad 1 \Big) z_{kj} + d_{i1} d_{i2} 2^{2m \ 1} \ z_{i1jm1} \quad z_{i2jm2} \\
&\quad + d_{i1} d_{i2} 2^{m1+m2 \ 2} \ z_{i1jm1} \quad z_{i2jm2} + d_i 2^{m+k \ 1} \ z_{ijm} \quad z_{kj} + 2^{k1+k2 \ 1} \ z_{k1j} \quad z_{k2j} \Big] ;
\end{aligned}
\tag{92}
$$

where if only one index is repeated we have $i1 < i2$, $m1 < m2$, $k1 < k2$, and if two are repeated we have $i1 \ , \ i2 \ \_ \ m1 \ , \ m2$.

The full problem Hamiltonian for the new setting will be the sum of all terms just mentioned,

$$
H_P = H_{\text{cost}} + H_u + H_g + H_t + H \quad + H \ : \tag{93}
$$

The number of qubits has increased by $J \sum_i (V_i + M_i)$, due to the codification of the new variables and the extra constraints we have been forced to introduced. Fortunately, the circuit schedule is the same, as the three ATA Hamiltonians that emerge from the new terms are equivalent to the ones that we had before: $\{q_{ij}; q_{ijm}; q_{ijv}\}$, $\{q`_j; q_j; q_{ij}\}$ and $\{q_{kj}; q_{ijm}\}$. Again, we can do this separation because all the interacting terms are of the form $^z \ ^z$, so that they always commute.

The discussion made on this section shows how we can add layers of complexity to our problem. In this regard, each time we add new a constraint or term to the function we are trying to optimize, we need to use more resources. The techniques developed in this work might help to construct new problems e ciently.

5. Numerical simulations and analysis

In this chapter, we present the results of our numerical experiments benchmarking the algorithm. First, we describe the algorithm used to perform the classical simulations. Then, we show the results of simulating the circuit solving the toy problem on an ideal quantum computer for di erent free input parameters, n_T, T and h_0. The results obtained in this chapter show the validity of our algorithm.

5.1 Classical simulation of the quantum circuit

The main problem we have to face when simulating a quantum system in a classical computer is the exponential amount of variables involved. For the case of simulating the evolution of a multiqubit system, the number of the variables to represent a pure state increases exponentially with the number of qubits, $O(2^N)$. For the matrix representation of the operators, the size increases as $O(2^{2N})$. The memory required to represent an arbitrary operator acting on 14 qubits is 4GB. When a new qubit is added, the memory involved increases by a factor of 4, imposing a limit on the size of systems that we can simulate.

The algorithm firstly prepares the initial state from the initial ground state $|0\rangle^{N}$. For this, a the Hadamard gate followed by a Z gate is applied to every qubit. Then, we have to simulate the circuit for the digitized evolution given by Eq. 71, which can be generated by calculating each of the exponential terms. The exponentials with H_0 are simulated with one SQG per qubit. To simulate the problem Hamiltonian H_P, we first divide it into local terms and the three ATA Hamiltonians which do not superpose shown in section 4.7. Since all terms pairwise commute, we can simulate them in any order. The local terms can be simulated by SQG rotations but, for the ATA Hamiltonians, we have to take first into account the kind of circuit topology involved. If we have a limited connectivity between the qubits, e.g. the realistic topology proposed in section 4.7, we have to implement the strategy from section 2.3.1. On the contrary, if we have an ideal topology, we can directly simulate the interacting terms with just one analog block per Hamiltonian, which can be implemented by employing an interaction time depending on their coupling strength. The steps following the algorithm for the simulation are shown in a more compact manner in algorithm 4.

However, this strategy is not e cient in terms of computational time. Each gate must be calculated as the tensor product of N 2×2 complex matrices, with N the number of qubits, and then multiplied by the vector representing the state. If we want to simulate an ideal circuit we can use a more e cient strategy. The initial state can be directly constructed by the tensor product of the ground state of σ^x, $((|0\rangle + |1\rangle)/\sqrt{2})^{N}$. Then, for the simulation of the evolution with H_0, we have that the evolution is identical for every qubit. Then, we can calculate the evolution for one qubit, and construct the evolution by taking the tensor product, $\exp(-i\,tH_0/\hbar) = \exp(-i\,th_0\,\sigma^x/\hbar)^{N}$. The evolution for H_P can be straightforwardly constructed employing the properties of the exponential of a matrix. Since H_P is a diagonal matrix, its exponential can be calculated as the exponential of the eigenvalues. These two small changes greatly improve the performance of the simulation, reducing the processor time needed by almost 3 orders of magnitude, from $1000s$ per time step to $1{:}5s$.

5.2 Numerical results

We have implemented and run a set of numerical experiments for this algorithm. Although we have implemented the improvements commented on the last paragraph of the previous section, the running time for each simulation is still high. On top of this, the memory required to simulate the circuit rapidly increases. The toy problem proposed in chapter 4 is the smallest non-trivial problem, consisting on a network of two factories and two clients. To solve the toy problem employing our algorithm, we have to simulate a 14-qubit circuit, and for that we need 4GB of RAM to store the matrix of H_0 and another 4GB for its evolution matrix. If we wanted to solve a more complex problem, the memory required would

Algorithm 4 Simulation of the circuit implementing the digitized QA for LND algorithm on a circuit with limited topology

Require: Hamiltonian parameters, number of steps n_T, annealing time T, scheduling function (t), qubit couplings

State preparation

1: Generate initial state, $|+\rangle = |0\rangle^{N}$
2: Initialize the state to the initial ground state, $|\psi\rangle = (ZH)^{N}|+\rangle$

Digitized evolution

3: Define the time step length, $t = T/nT$
4: **for** each time step, k = 1:nT-1 **do**
5: Evolve the state with H_0, $|\psi\rangle = RX((1 \quad (k \quad t))\ th_0/\sim)^{N}|\psi\rangle$. Skip this step for k=1
6: Evolve the state with the local terms of H_P, $|\psi\rangle = \prod_i^N RZ(2 \quad (k \quad t) \quad t(\text{local term i})/\sim)|\psi\rangle$
7: **for** each ATA Hamiltonian group $\{\{j, j; i, j\}_j; \{i, j; k, j\}_j; \{i, j\}_i\}$ **do** . See section 4.7
8: Simulate the evolution of the ATA Hamiltonian from each separate subspace within the group in parallel using the algorithm from section 2.3.1
9: Evolve the state with H_0, $|\psi\rangle = RX((1 \quad (k \quad t))\ th_0/\sim)^{N}|\psi\rangle$

Measurement on the computational basis

10: Obtain the probability of measuring the state in each of the states of the computational basis $\{|\quad_m\rangle\}$, $p_m = |\langle \quad_m|\quad\rangle|^2$
11: Randomly choose one of the states according to their probability distribution p_m, $|\quad_{\text{meas}}\rangle$
12: Discard the information about the ancilla qubits, $|\quad_{\text{meas}}\rangle = |\quad_{\text{sol}}\rangle \quad |\quad_{\text{ancilla}}\rangle$
13: Extract the information of the solution for the LND problem, $|\quad_{\text{sol}}\rangle \ ! \ x_j; y_{i;j}$. See section 4.1

increase too much to be simulated in a regular computer. For instance, a problem consisting on three factories and two clients needs 21 qubits, and in order to store a complex matrix of size $2^{21} \quad 2^{21}$ we need 64TB. There are ways of overcome this problem, however, the running time of such programs would completely una ordable. For these reasons, we have analysed the algorithm against the toy problem proposed in chapter 4 in detail.

We have implemented the algorithms described in this chapter on *Matlab 2021a*. To run the programs, we have used three computers: one where we have made the benchmarks on with a *AMD 3600X* @4.0GHZ CPU a with 16GB of RAM and a *RTX 2700 Super* graphics card with 10GB of VRAM, and two computers with a *Intel Xeon E5* @2.7GHz with 64GB of RAM. Whenever it was possible, we have used the *Parallel Computing Toolbox* package and the graphics card to speed up the calculations.

5.2.1 Annealing time and number of steps

As discussed in section 4.8, the relation between the annealing time (T) and the number of steps into which the evolution is divided (n_T) is directly connected with the fidelity of the algorithm. To gain some intuition about this relation, we have launched a series of simulations in which we have varied T and n_T. It is key to know the region in which our algorithm performs with an acceptable fidelity. Thus, we have also searched for the n_T that achieves certain fidelity for di erent T's. This has been done with the binary search or bisection algorithm [52, Chapter 6.1.1]. The results of these calculations are shown in Fig. 12.

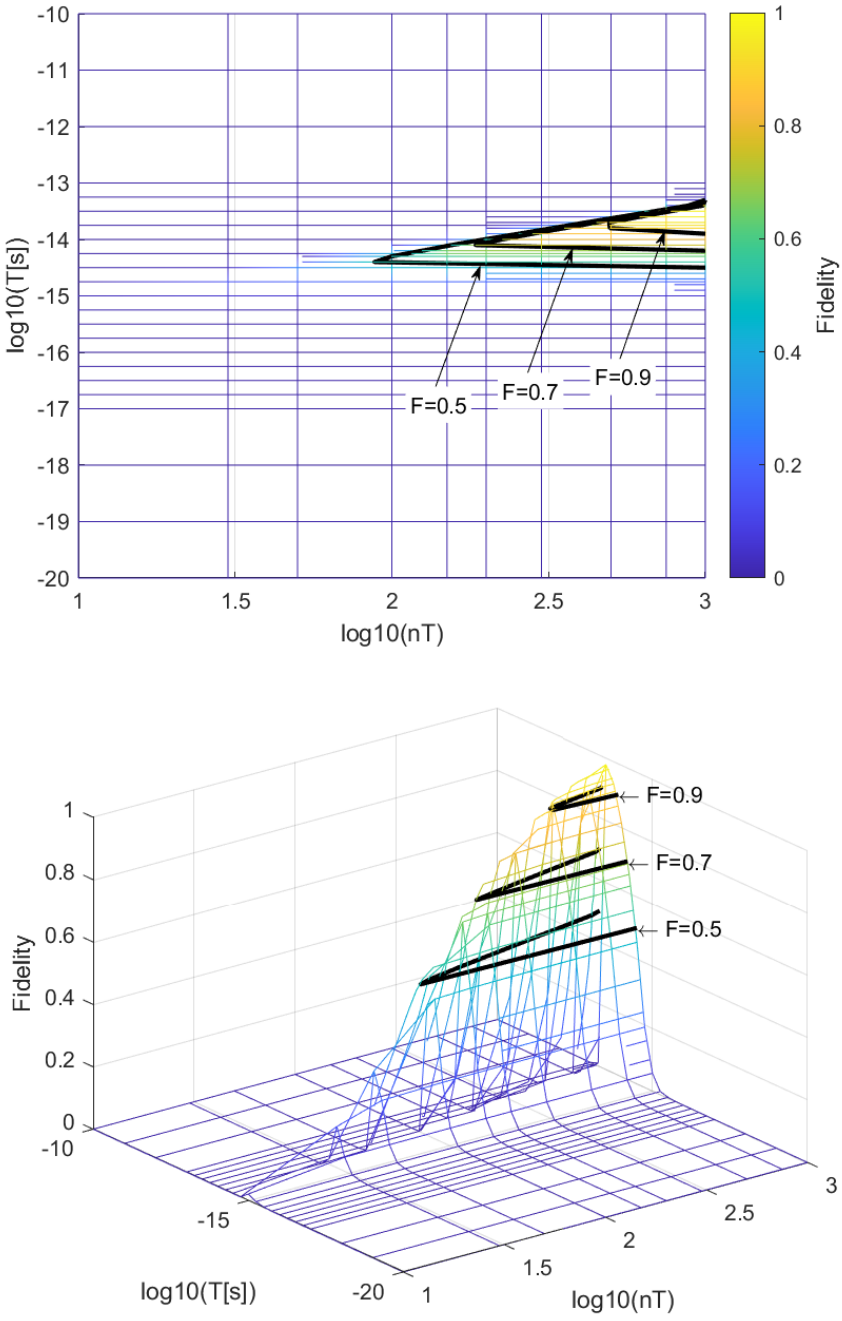


Fig. 12: Plot of the delity vs the logarithm of the annealing time T and the number of steps for the discretization n_T for the toy problem. The thick black lines mark the contour of the surface that intersects with a delity of $F = 0{:}5$, $F = 0{:}7$ and $F = 0{:}9$. We observe two di erent behaviours at both sides of the surface. Going from lower to greater annealing times, we observe that the delity rapidly grows from a point around $T_{\min}$ $10^{-15}s$. This behaviour corresponds to the growth in delity of the annealing process, which solely depends on the annealing time. On the other side of the surface, the delity drops almost vertically at the limit de ned around $\log_{10}(T[s])$ $1{:}00\log_{10}(nT)$ $16{:}39$. The appearance of this limit is consistent with the discussion about the digitization error from section 4.8. This gives us an upper limit for the length of a time step for the di erent delity values of $t_{\max}(F = 0{:}5)$ $4{:}8$ $10^{-17}s$, $t_{\max}(F = 0{:}7)$ $4{:}4$ $10^{-17}s$ and $t_{\max}(F = 0{:}9)$ $4{:}0$ $10^{-17}s$.

The most important conclusion extracted from this numerical experiment is a confirmation that the algorithm successfully solves the problem given su ciently annealing time and number of steps. Indeed, although not fully verified due to the computational cost of the calculation, it seems that the fidelity of the simulation converges to a fidelity close to 1 for su ciently large T and n_T. In fact, we have achieved a maximum fidelity of 0.978 for an annealing time of $T = 10^{13:5}$ s and $n_T = 1000$. In section 2.2 we observed that the annealing process converge to the exact problem Hamiltonian ground state given a su ciently long annealing time. Additionally, we have also noticed in sections 2.5 and 4.8 that the digitized annealing process converges to its continuous counterpart for su cient number of steps. Then, it is a sensible conclusion to assume that our algorithm also converges to a fidelity 1 in these limits.

For a very large n_T, we expect that the fidelity of the simulation decreases due to the accumulation of the representation error. As we are using a binary representation for real numbers, we have a limitation in the precision. The precision varies depending on the order of magnitude of the represented number, which is about 14 orders of magnitude below of it. Fortunately, the number of steps needed so that the representation error is relevant, is too large to be taken into account. Even if this were the case, the problem could be solved by simply extending the representation format of real numbers from double-precision (64 bits) to quadruple-precision (128 bits).

The results also provide a method for choosing the annealing time and number of steps to achieve a reasonable fidelity for a single run. To further increase the fidelity, we can run again the algorithm. The probability of obtaining the correct result for a problem after a number of di erent runs given a fidelity grows as

$$P(F; \#\text{runs}) = 1 \quad (1 \quad F)^{\#\text{runs}}: \tag{94}$$

Then, we notice that the number of runs to obtain the result with a great confidence is relatively small. For a circuit with a fidelity of 0.7 we would measure the correct result at least once with a confidence of 99.76% just after five runs, and with 99.9994% confidence after 10 runs.

5.2.2 h_0 and the gap

As we have discussed in section 4.3, the h_0 constant is a free parameter in the problem. Although at first glance, this parameter seems to lack physical meaning, it directly a ects the fidelity in di erent modes.

On the one hand, we have to take into account that the initial gap for the evolution is given precisely by $E = 2h_0$. As we have seen in section 2.2, the smaller the minimum gap, the worse the fidelity. Then, if we set h_0 too small, the fidelity will drop in both QA and DQA algorithms.

On the other hand, the norms of the Hamiltonians are relevant for the approximations made when discretizing the evolution and the truncation of the Trotter expansion. In the discussion about the errors in section 4.8, we obtained that the discretization error is negligible if $\|H_0 \quad H_P\| \quad (t=2\sim nT)\|[H_0; H_P]\|$ and that the Trotterization error grows with $(t=\sim)^3\|[H_0; H_P]\|$. For the limit $\|H_0\| = Nh_0 \quad \|H_p\|$, we have that the discretization error is independent of h_0, $h_0 \quad th_0\|H_P\| = \sqrt{2}\sim n_T \quad ! \quad N \quad t\|H_P\| = \sqrt{2}\sim nT$. However, this is not the case for the Trotterization error, which in this limit grows as $\sqrt{2N} \quad t^2h_0\|H_P\| = \sim^2$. To obtain this last expression, we have used the upper bound for the norm of the commutator given by Böttcher and Wenzel [53].

To verify these assumptions, we have calculated the fidelity for di erent values of $h_0 = \{1; 3:07; 30\}$, two di erent step numbers and annealing times. The results are depicted in Fig. 13. We could have performed an extensive analysis for more values of h_0 and n_T. Nevertheless, the numeric experiments carried out, show that the fidelity for an intermediate value of h_0 is indeed greater than the maximum fidelity for values at the extremes.

Figure 14 shows the evolution of the gap between the ground and the first excited states for di erent values of h_0. As we expected, the minimum gap increases with h_0. A further conclusion we extract is

that the time for the minimum gap shifts also with h_0. This might be useful for further optimizing the annealing scheduling, which might consist of increasing the speed at which we approach and leave the minimum gap point.

If we extend the simulations to reproduce Fig. 12 for di erent values of h_0, we expect to obtain similar results. As we see from Fig. 13, the fidelity increases almost linearly from a certain value of the annealing time, which does not depend on the number of steps employed. For larger values of T, the fidelity abruptly drops. However, for a larger number of steps this occurs for larger values of T. Since we have not calculated a su cient number points for computational restrictions, we can not give the contour of the surface cutting the plane for a constant fidelity value. Nevertheless, we are confident that the resulting shape of the surface will be similar for the three cases for the aforementioned physical reasons.

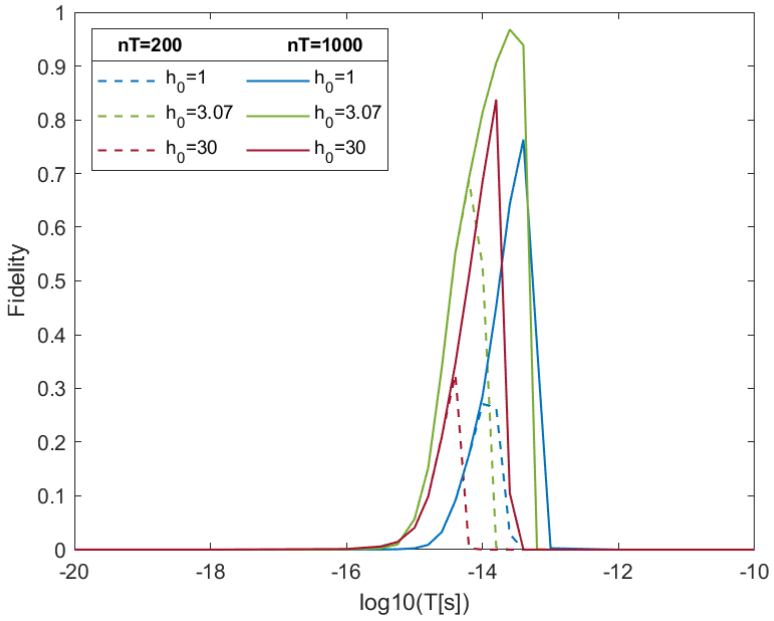


Fig. 13: Fidelity for di erent values of h_0 and n_T vs the annealing time T. Here, we observe that the delity for a xed n_T corresponds to the general behaviour deduced in this section. The delity of the algorithm for a small $h_0 = 1$ and large $h_0 = 30$ are lower than the delity for an intermediate value $h_0 = 3{:}07$. The expected tendency is to show a maximum delity between the two extremes. For smaller h_0, the delity drops due to the small gap at the start of the evolution schedule; while for larger h_0, the Trotterization error increases with $\|H_0\| = Nh_0$, as discussed in section 4.8.

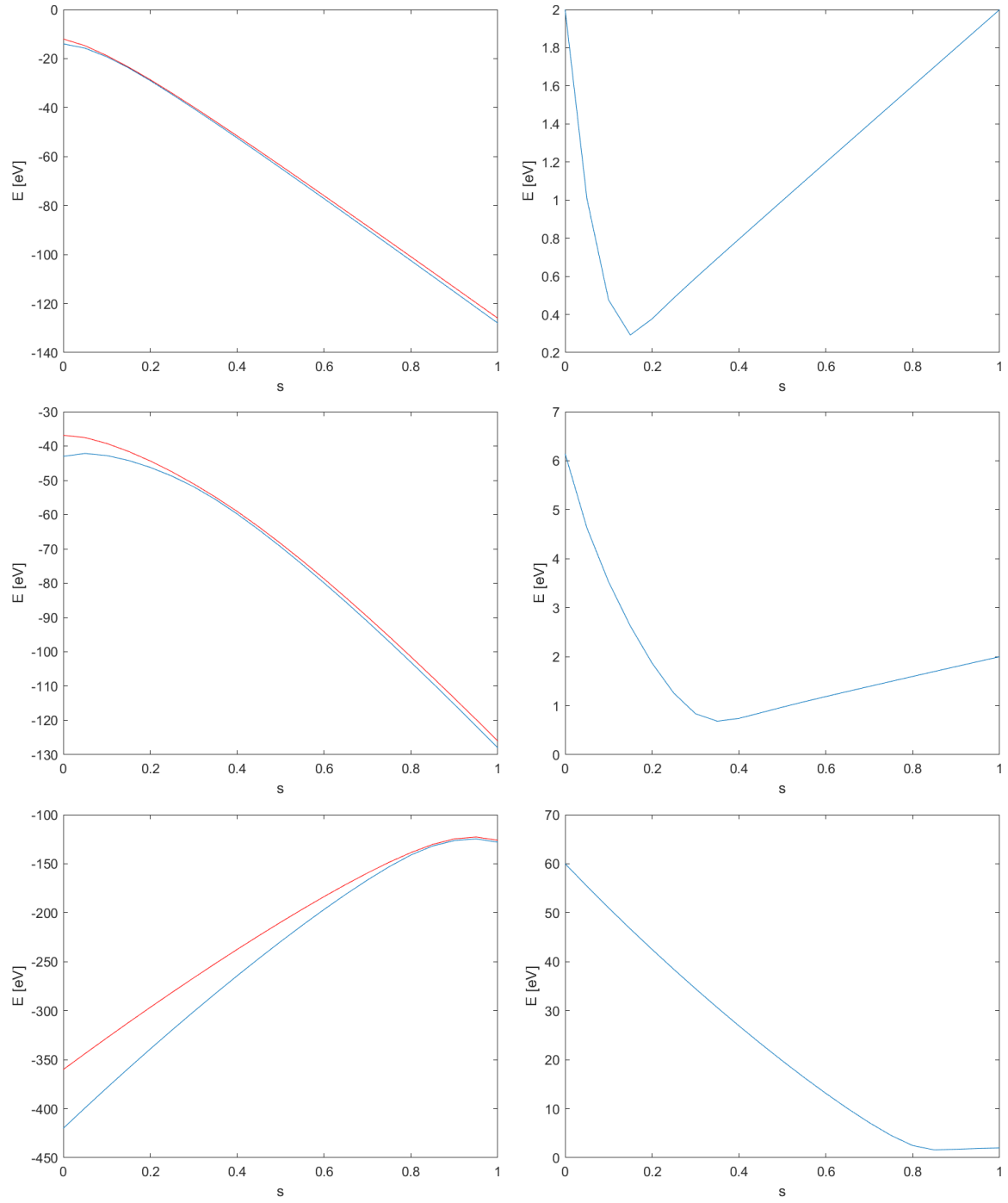


Fig. 14: Evolution of the ground state and the rst excited state of the system for the toy problem Hamiltonian and di erent values of h_0. From top to bottom we plot $h_0 = 1$, $h_0 = 3{:}07$ and $h_0 = 30$. The plots on the left show the evolution in terms of the absolute energy, while in the plots at right, we show the gap between these two states. The x axis is given in terms of an adimensional parameter s, representing the fraction of the evolution $s = t/T$. This choice allows us to study the evolution of the gap without xing the annealing time T beforehand. We depict here the results for 21 s ticks evenly separated. The annealing scheduling function here is $(t) = t/T$, which in terms of the parameter s is precisely $(s) = s$. The minimum gap for each h_0 is $E(h_0 = 1) = 0{:}3$eV, $E(h_0 = 3{:}07) = 0{:}7$eV and $E(h_0 = 30) = 1{:}6$eV.

6. Conclusion

In this thesis, we present the digitized version of QA addressing the LND problem, i.e. the optimization of the cost for the configuration of factories and supply lines supplying a distribution of clients. Starting from various previous works, we design a digitized algorithm that successfully codifies a LND problem and its constraints into a Hamiltonian, whose ground state codifies the solution to the problem. For that, we introduce an optimized number of ancilla qubits. Via a digitized quantum annealing process, we evolve the easily preparable ground state of the system. The digitized annealing process assures that the system remains in its ground state at every point of the evolution with high probability, in particular at the end of the process. Then, the information about the solution is extracted by discarding the ancilla qubits and measuring the system in the canonical basis.

Regarding the problem codification, we successfully codify all constraints for the LND problem. We also improve previous approaches by giving a new strategy to select the penalty factors applied to the states which do not fulfill the constraints. The new selection of parameters generates a Hamiltonian with a lower norm than previously achieved. This fact reduces the error introduced into the digitization process, improving the fidelity of the algorithm.

We have also studied the physical realization of quantum computers, which enables us to give the optimal topology of a realistic quantum system which could implement our algorithm. For that, we give the tools to implement the algorithm in the DAQC paradigm on a system with the connections of various NN Hamiltonians that overlap, forming a square lattice.

Although we have solved one LND problem, the algorithm we propose can be applied to other constrained optimization problems. As an example of this, we propose a different LND problem with loose constraints, i.e. with a more complex cost function.

To benchmark the algorithm, we simulate the smallest non-trivial problem due to hardware limitations. Indeed, we address an LND problem, comprising two factories and two clients. The results show that the fidelity grows when the number of steps for the discretization increases. Regarding the annealing time, we find a minimum annealing time for the fidelity to reach relevant values. Similarly, we also find that there is a maximum length for the time steps from which the fidelity rapidly decreases. The results obtained are consistent with the theoretical predictions.

In order to challenge the protocol, it would be necessary to scale up the size of the problem. For this goal, we would need to either find a method to reduce the memory to store the matrix representations of the operators, or to have access to sufficiently large quantum processors.

If we additinaly want to keep the accuracy under control while scaling up the problem, we need to mitigate the effect of noise on our algorithm. For that, we could follow the work of García-Molina et al. [54], where they introduce quantum error mitigation techniques in the digital-analog framework. In that work, the authors show that the advantage of DAQC over DQC appears when considering the noise. They also conclude that the fidelity for the stepwise DAQC (sDAQC) we have used in this thesis is lower than the one obtained with the banged DAQC (bDAQC). This new paradigm leaves the interactions turned on at every time, applying the SQG on top of them. Although bDAQC does not implement the exact evolution expected from DQC and sDAQC, it substantially reduces the noise of turning on and off the interactions. The reduction of the noise compensates for the mentioned difference in the evolution. We expect to obtain similar advantages when implementing our algorithm in a NISQ processor.

To sum up, we have designed a quantum algorithm for LND which outperforms previous quantum approaches in several aspects. However, in order to provide a functional quantum algorithm, much further research is required. We have identified multiple possible improvements that will expectedly enhance the applicability of the algorithm, paving the way for useful application of quantum computing in logistics.

In addition to the aforementioned pathways to improve our algorithm, we could further improve it by introducing shortcuts to adiabaticity techniques (STA) in the initial annealing process [10]. In particular, we would realize this by introducing counter-diabatic (CD) terms. Searching for the exact CD term to optimize the evolution is a di cult task, even for simple problems like Ising Hamiltonians. To overcome this problem, we can search for approximate solutions [55] or restrict ourselves to only finding local CD terms [56].

References

[1] J. Kleinberg and E. Tardos, *Algorithm design* (Addison-Wesley Longman Publishing Co. Inc., USA, 2005).

[2] P. A. Benio , "Quantum mechanical Hamiltonian models of discrete processes that erase their own histories: application to Turing machines", International Journal of Theoretical Physics **21**, 177–201 (1982).

[3] R. P. Feynman, "Simulating physics with computers", International Journal of Theoretical Physics **21**, 467–488 (1982).

[4] L. K. Grover, "A fast quantum mechanical algorithm for database search", in Proceedings of the twenty-eighth annual acm symposium on theory of computing, STOC '96 (1996), pp. 212–219.

[5] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring", in Proceedings 35th annual symposium on foundations of computer science (1994), pp. 124–134.

[6] E. K. Grant and T. S. Humble, "Adiabatic quantum computing and quantum annealing", Oxford Research Encyclopedia of Physics (2020).

[7] B. Tamir and E. Cohen, "Notes on adiabatic quantum computers", arXiv:1512.07617 (2016).

[8] A. Y. Kitaev, "Quantum computations: algorithms and error correction", Russian Mathematical Surveys **52**, 1191–1249 (1997).

[9] M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, "Experimental realization of any discrete unitary operator", Physical Review Letters **73**, 58–61 (1994).

[10] N. N. Hegade, K. Paul, Y. Ding, M. Sanz, F. Albarrán-Arriagada, E. Solano, and X. Chen, "Shortcuts to adiabaticity in digitized adiabatic quantum computing", Phys. Rev. Applied **15**, 024038 (2021).

[11] M. Grajcar, A. Izmalkov, S. H. W. van der Ploeg, S. Linzen, T. Plecenik, T. Wagner, U. Hübner, E. Il'ichev, H.-G. Meyer, A. Y. Smirnov, P. J. Love, A. Maassen van den Brink, M. H. S. Amin, S. Uchaikin, and A. M. Zagoskin, "Four-qubit device with mixed couplings", Phys. Rev. Lett. **96**, 047006 (2006).

[12] S. Hazra, K. V. Salunkhe, A. Bhattacharjee, G. Bothara, S. Kundu, T. Roy, M. P. Patankar, and R. Vijay, "Engineering cross resonance interaction in multi-modal quantum circuits", Applied Physics Letters **116**, 152601 (2020).

[13] X. Gu, A. F. Kockum, A. Miranowicz, Y. X. Liu, and F. Nori, "Microwave photonics with superconducting quantum circuits", Physics Reports **718-719**, Microwave photonics with superconducting quantum circuits, 1–102 (2017).

[14] J. Yu, J. C. Retamal, M. Sanz, E. Solano, and F. Albarrán-Arriagada, "Superconducting circuit architecture for digital-analog quantum computing", arXiv:2103.15696 (2021).

[15] *Ibm quantum*, https://quantum-computing.ibm.com, 2021.

[16] P. I. Bunyk, E. M. Hoskinson, M. W. Johnson, E. Tolkacheva, F. Altomare, A. J. Berkley, R. Harris, J. P. Hilton, T. Lanting, and A. J. Przybysz, "Architectural considerations in the design of a superconducting quantum annealing processor", IEEE Transactions on Applied Superconductivity **24**, 1–10 (2014).

[17] P. V. Klimov, J. Kelly, Z. Chen, M. Neeley, A. Megrant, B. Burkett, R. Barends, K. Arya, B. Chiaro, Y. Chen, A. Dunsworth, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Gra , T. Huang, E. Je rey, E. Lucero, J. Y. Mutus, O. Naaman, C. Neill, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, S. Boixo, R. Babbush, V. N. Smelyanskiy, H. Neven, and J. M. Martinis, "Fluctuations of energy-relaxation times in superconducting qubits", Phys. Rev. Lett. **121**, 090502 (2018).

[18] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits", Applied Physics Reviews **6**, 021318 (2019).

[19] D. Willsch, M. Nocon, F. Jin, H. Raedt, and K. Michielsen, "Gate error analysis in simulations of quantum computers with transmon qubits", Physical Review A **96**, 062302 (2017).

[20] S. Ball, A. Centelles, and F. Huber, "Quantum error-correcting codes and their geometries", arXiv: 2007.05992 (2020).

[21] A. Bienfait, J. J. Pla, Y. Kubo, X. Zhou, M. Stern, C. C. Lo, C. D. Weis, T. Schenkel, D. Vion, D. Esteve, J. J. L. Morton, and P. Bertet, "Controlling spin relaxation with a cavity", Nature **531**, 74–77 (2016).
[22] K. Khodjasteh and D. A. Lidar, "Fault-tolerant quantum dynamical decoupling", Phys. Rev. Lett. **95**, 180501 (2005).
[23] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing", Science **220**, 671–680 (1983).
[24] M. Born and V. Fock, "Beweis des Adiabatensatzes", Zeitschrift für Physik **51**, 165–180 (1928).
[25] T. Kato, "On the adiabatic theorem of quantum mechanics", Journal of the Physical Society of Japan **5**, 435–439 (1950).
[26] M. H. S. Amin, "Consistency of the adiabatic theorem", Physical Review Letters **102**, 220401 (2009).
[27] S. Boyd and L. Vandenberghe, *Convex optimization* (Cambridge University Press, 2004).
[28] A. Galicia, B. Ramon, E. Solano, and M. Sanz, "Enhanced connectivity of quantum hardware with digital-analog control", Phys. Rev. Research **2**, 033103 (2020).
[29] B. Alspach, "The wonderful Walecki construction", Bulletin of the Institute of Combinatorics and its Applications **52**, 7–20 (2008).
[30] Y. Sun, J. Y. Zhang, M. S. Byrd, and L. A. Wu, "Adiabatic quantum simulation using trotterization", arXiv:1805.11568 (2018).
[31] H. F. Trotter, "On the product of semi-groups of operators", Proceedings of the American Mathematical Society **10**, 545–551 (1959).
[32] M. Suzuki, "Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems", Communications in Mathematical Physics **51**, 183–190 (1976).
[33] S. Masuo, "Decomposition formulas of exponential operators and Lie exponentials with some applications to quantum mechanics and statistical physics", Journal of Mathematical Physics **26**, 601–612 (1985).
[34] D. Poulin, A. Qarry, R. Somma, and F. Verstraete, "Quantum simulation of time-dependent Hamiltonians and the convenient illusion of Hilbert space", Physical Review Letters **106**, 10.1103/physrevlett.106.170501 (2011).
[35] Y. Ding, X. Chen, L. Lamata, E. Solano, and M. Sanz, "Implementation of a hybrid classical-quantum annealing algorithm for logistic network design", SN Computer Science **2**, 68 (2021).
[36] N. C. Dzupire and Y. Nkansah-Gyekye, "A multi-stage supply chain network optimization using genetic algorithms", arXiv:1408.0614 (2014).
[37] M. Gen, L. Lin, and J.-B. Jo, "Hybrid genetic algorithm for designing logistics network, vrp and agv problems", in *Intelligent and evolutionary systems* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2009), pp. 123–139.
[38] L. Lin, M. Gen, and X. Wang, "Integrated multistage logistics network design by using hybrid evolutionary algorithm", Computers & Industrial Engineering **56**, 854–873 (2009).
[39] C. Silva, J. Sousa, T. Runkler, and J. Costa, "Distributed supply chain management using ant colony optimization", European Journal of Operational Research **199**, 349–358 (2009).
[40] T. G. Chen and C. H. Ju, "A novel artificial bee colony algorithm for solving the supply chain network design under disruption scenarios", International Journal of Computer Applications in Technology **47**, 289–296 (2013).
[41] N. Hiremath, S. Sahu, and M. Tiwari, "Designing a multi echelon flexible logistics network using co-evolutionary immune-particle swarm optimization with penetrated hypermutation (coipso-phm)", Applied Mechanics and Materials **110-116**, 3713–3719 (2011).
[42] W. Wang and W. Wang, "Logistics network design problem with information sharing strategy", in 2011 international conference on management and service science (2011), pp. 1–4.
[43] J. Qin and L. X. Miao, "Combined simulated annealing algorithm for logistics network design problem", in 2009 international workshop on intelligent systems and applications (2009), pp. 1–4.

[44] X. Wan, J. F. Pekny, and G. V. Reklaitis, "Simulation-based optimization with surrogate models - application to supply chain management", Comput. Chem. Eng. **29**, 1317–1328 (2005).
[45] B. Dury and O. D. Matteo, "A qubo formulation for qubit allocation", arXiv:2009.00140 (2020).
[46] T. Albash and D. A. Lidar, "Adiabatic quantum computation", Reviews of Modern Physics **90**, 015002 (2018).
[47] Y. Q. Chen, Y. Chen, C. K. Lee, S. Zhang, and C. Y. Hsieh, "Optimizing quantum annealing schedules: from monte carlo tree search to quantumzero", (2020).
[48] R. Barends, A. Shabani, L. Lamata, J. Kelly, A. Mezzacapo, U. L. Heras, R. Babbush, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, E. Je rey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, E. Solano, H. Neven, and J. M. Martinis, "Digitized adiabatic quantum computing with a superconducting circuit", Nature **534**, 222–226 (2016).
[49] G. B. Mbeng, R. Fazio, and G. E. Santoro, "Optimal quantum control with digitized quantum annealing", arXiv:1911.12259 (2019).
[50] J. B. Altepeter, D. Branning, E. Je rey, T. C. Wei, P. G. Kwiat, R. T. Thew, J. L. O'Brien, M. A. Nielsen, and A. G. White, "Ancilla-assisted quantum process tomography", Physical Review Letters **90**, 193601 (2003).
[51] A. Lupaşcu, S. Saito, T. Picot, P. C. de Groot, C. J. P. M. Harmans, and J. E. Mooij, "Quantum non-demolition measurement of a superconducting two-level system", Nature Physics **3**, 119–123 (2007).
[52] P. Scherer, *Computational physics. Simulation of classical and quantum systems* (Springer-Verlag Berlin Heidelberg, 2010).
[53] A. Böttcher and D. Wenzel, "The frobenius norm and the commutator", Linear Algebra and its Applications **429**, 1864–1885 (2008).
[54] P. García-Molina, A. Martin, and M. Sanz, "Noise in digital and digital-analog quantum computation", arXiv:2107.12969 (2021).
[55] T. Hatomura, "Shortcuts to adiabaticity in the infinite-range ising model by mean-field counter-diabatic driving", Journal of the Physical Society of Japan **86**, 094002 (2017).
[56] D. Sels and A. Polkovnikov, "Minimizing irreversible losses in quantum systems by local counter-diabatic driving", Proceedings of the National Academy of Sciences **114**, E3909–E3916 (2017).