Optimization Linear Apply in Quantum Computer

In the previous article (https://www.linkedin.com/pulse/introduction-quantum-computing-building-bit-array-circuit-eduardo/) we show an overview on quantum computers, in this article we will describe a linear optimization problem using algorithm and then introduce using a quantum computer D-Wave System (https://www.dwavesys.com/), D-Wave said to be the first company in the world to sell computers that exploit quantum effects in their operation , although some researchers disagree, it takes a different approach on the quantum computers of port models, where they use the quantum anneling idea to solve optimization problems.

The quantum anneling is a shifted state system that is gradually linked to the problem where quantum physics accomplishes these changes, the configuration at the end corresponds to the response we are trying to find, it is implemented as a single quantum algorithm, and this scalable approach. Currently, the QPU D-Wave 2000Q supports a chimera graph with its 2048 qubits that are logically mapped on a 16x16 unit cell of 8 qubits, Figure 1 shows a piece of that graph.

Each unit cell can have two formats which is the cross and columnar, Figure 2 shows both formats.

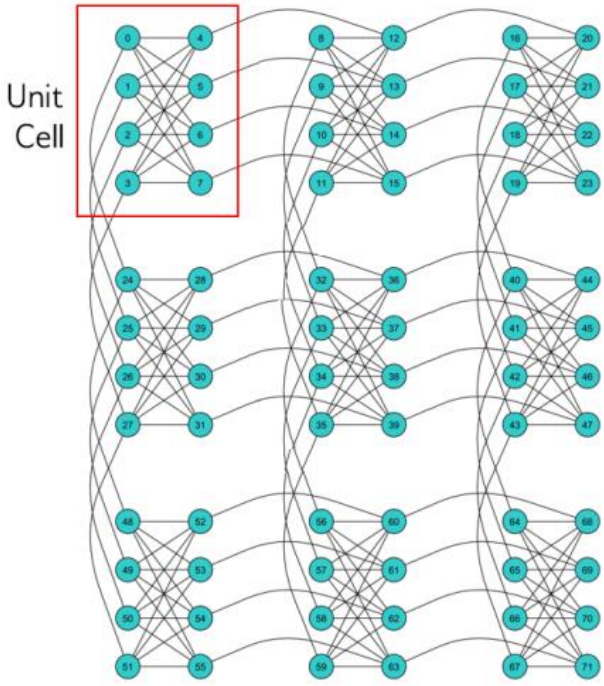

Figure 1 – Illustration of a 3x3 chimera graph.

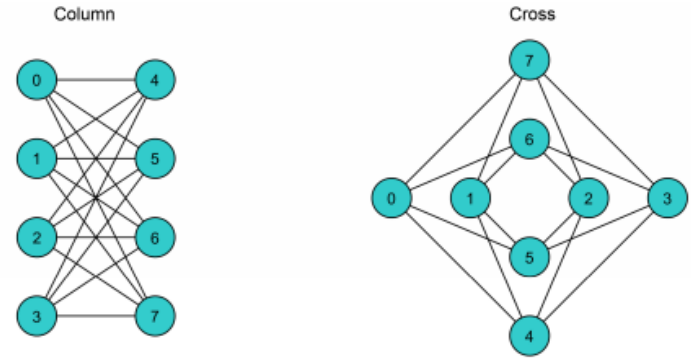

Figure 2 – Types of qubits formats in each cell unit of the chimera graph

In D-wave there are two types of formulations that is Ising and QUBO, let each one of them:

- Ising

  The Ising model of physicist Ernest Ising, is a mathematical model traditionally used in statistical mechanics, the model consists of discrete variables that represent atomic spins that can be in one of the two states (+1 or -1). Spins are organized on a graph, usually a network, allowing each spin to interact with its neighbors. The model allows the identification of phase transitions, as a simplified model of reality, equation [1] illustrates this mathematical model.

$$E_{ising}(s) = \sum_{i=0}^{N} h_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{i,j} s_i s_j$$

  where the linear coefficients corresponding to the vies of the qubit that are h_i, and the quadratic coefficients corresponding to the coupling forces are $J_{i,j}$.

- QUBO

  QUBO problems are traditionally used in computer science. Variables are TRUE and FALSE, states that correspond to values 1 and 0. A QUBO problem is defined using an upper diagonal matrix Q, which is a triangular matrix N x N of actual weights, a vector of binary variables, as minimizing the function [2].

$$f(x) = \sum_{i} Q_{i,i}\, x_i + \sum_{i<j} Q_{i,j} x_i x_j$$

  where the diagonal terms $Q_{i,i}$ are the linear coefficients and the non-diagonal terms are the quadratic coefficients $Q_{i,j}$.

In our approach we will use the QUBO model and we will compare the result with the Simplex technique, which is an algorithm created by the mathematician George Dantzig that makes possible the solution of many linear programming problems. Quite popular, it finds good acceptance in areas where diverse needs and constraints influence a value that needs to be increased or decreased to the maximum. Formally, the complexity of the Simplex algorithm is taken as exponential [1] shows that, in a naive implementation, each iteration in search of the best solution has, in principle, complexity $O(nm^4)$ in terms of m variables and n restricts. constraints. However, with the linear transformation approach where at each iteration the whole system is transformed so that the previous vertex is the new origin, the complexity by iteration becomes $O(mn)$.

Otimização combinatória

Combinatorial Optimization is a branch of computer science and applied mathematics that studies optimization problems in finite sets.

In an optimization problem we have an objective function and a set of constraints, both related to the decision variables. The possible values for the decision variables are delimited by the constraints imposed on these variables, forming a discrete set (finite or not) of feasible solutions to a problem. The problem can be minimization or maximization of the objective function. The answer to the optimization problem, that is, the global optimum, will be the smallest (or largest) possible value for the objective function for which the value assigned to the variables does not violate any constraint. In some cases, we reach values whose discrete change does not lead to better results, but which are not also the global optimum - to these solutions we call the optimal location.

There are many possible classifications for the optimization problem, and some of them will present accurate and efficient methods of resolution. Others will lead to the need for non-exact (heuristic) methods, since their exact formulation and/or resolution would lead to intractable complexity.

As techniques of exact solutions - especially with polynomial algorithm for some problems - we have, for example:

- Graphs of algorithms(Graphs of Theory)
- Greedy algorithms
- **Algorithm simplex**
- Branch and bound
- Dynamic programming
- Lagrangian Relaxation
- Programming with constraints

Some of the techniques for obtaining approximate solutions:

- Genetic algorithms
- Search Tabu
- Ant Colony Algorithm
- Greedy Randomized Adaptive Search Procedures (GRASP)
- Neural Networks
- Simulated annealing

The variables can assume values of type:

- Real programming (unused name, just put here for differentiation)
- Whole programming
- **Programming 0-1 (integer variables, with only two possible values)**
- Mixed programming (some real and other whole variables)

The objective function of the problem assumed two types of function are:

- Convex function - Great location is global (simpler)
- Concave function - Great location not necessarily Global (more complicated to solve)

We will approach the simplex technique in comparison to the annealing algorithm using the D-wave quantum machine, below the proposed problem of discrete linear optimization.

$$\max 7x_1 + 4x_2 + 19x_3$$

$$sa\ x_1 + x_3 \leq 1$$

$$x_2 + x_3 \leq 1$$

$$x_1, x_2, x_3 = 0\ ou\ 1$$

According to Table 1 we have the solutions and the values of the objective function for each solution,

Solution	Goal
(0,0,0)	0
(0,0,1)	**19**
(0,1,0)	4
(0,1,1)	Impracticable
(1,0,0)	7
(1,0,1)	Impracticable
(1,1,0)	11
(1,1,1)	Impracticable

Table 1 – All solutions and their results

Looking at Table 1 we see that the highest value obtained was 19 of the solution (0,0,1), with this example we set up an application in Python using the scipy library with the class optimize using the linprog function to apply the simplex in this problem and for application on the D-Wave computer we use their SDK, where it can be installed via pip using this "pip install dwave-ocean-sdk" command or by performing the git clone of this link (https://github.com/dwavesystems/dwave-ocean-sdk.git) and perform the "python setup.py install", the setup is necessary for the user to register on the site (https://cloud.dwavesys.com/leap/) to generate a token to free access to the machine, with this you will have one minute of access to test some optimization applications.

Any solution developed to solve our proposed problem is available at this link (https://github.com/valteresj2/Quantum-Solution-Linear-Optimization) of github, below I will comment on some important functions to transform an optimization problem linear relationship in a quadratic problem to fit the D-Wave computer model that works with quantum entanglement using the Chimera graph shown in Figures 1 and 2.

The first transformation to be made is to transform the in questions into equations. The solution addressed will be to add auxiliary variables and then the linear constraints will be transformed into quadratic terms.

$$1.\,(x_1 + x_3 + s_1 - 1)^2 = 0$$
$$2.\,(x_2 + x_3 + s_2 - 1)^2 = 0$$

The product_notable function performs the notables products for each constraint and we get:

$$1.\,x_1^2 + 2x_1x_3 + 2s_1x_1 + 2s_1x_3 + x_3^2 + s_1^2 - 2x_1 - 2x_3 - 2s_1 + 1 = 0$$
$$2.\,x_2^2 + 2x_2x_3 + 2s_2x_2 + 2s_2x_3 + x_3^2 + s_2^2 - 2x_2 - 2x_3 - 2s_2 + 1 = 0$$

As the variables are dichotomous, we assume that $x_1^2 = x_1$, $x_2^2 = x_2$, $x_3^2 = x_3$, $s_1^2 = s_1$ and $s_2^2 = s_2$ so we have the following restrictions:

$$1.\,-x_1 - x_3 - s_1 + 2x_1x_3 + 2x_1s_1 + 2x_3s_1 + 1 = 0$$
$$2.\,-x_2 - x_3 - s_2 + 2x_2x_3 + 2x_2s_2 + 2x_3s_2 + 1 = 0$$

Applying the sum of the constraints $\sum_{i=1}^{n} R_i$, where n is the number of problem constraints, we have:

$$(-x_1 - x_3 - s_1 + 2x_1x_3 + 2x_1s_1 + 2x_3s_1 + 1) + (-x_2 - x_3 - s_2 + 2x_2x_3 + 2x_2s_2 + 2x_3s_2 + 1)$$

At the end the product_notable function returns the final equation [1],

$$(-x_1 - x_2 - 2x_3 - s_1 - s_2 + 2x_1x_3 + 2x_2x_3 + 2x_1s_1 + 2x_3s_1 + 2x_2s_2 + 2x_3s_2 + 2)\ [1]$$

With the final equation [1] we can assemble our chimera graph that illustrates the links between their variables with their respective constants, for our problem we have 5 variables and each variable is a qubit, remembering that this is just a learning demo the ideal is for problems with a high density (volume) of variables.

The model_dwave function transforms the result of the 'product_notable' function into the format for the 'sample_qubo' function that is in the D-Wave package to run and bring the result of possible solutions that fall into the user-determined rounds.

Another relevant function is the 'result_dwave' it receives the results of the 'sample_qubo' function that is selected the one that presents the lowest energy and if it still comes more than one possible solution applies these solutions in the objective function in order to bring the best solution that maximize or minimize the objective function.

Reference:

[1] S. Dasgupta, C. H. Papadimitriou and U. V. Vazirani, “Algorithms”, McGraw-Hill, Boston, 2006.