COMMUNITY DETECTION USING QUANTUM COMPUTERS

CSE622: INTRODUCTION TO QUANTUM COMPUTING
WINTER SEMESTER, 2020

INSTRUCTOR: DR. DEBAJYOTI BERA

PROJECT REPORT BY

CHAITANYA AGRAWAL (2016027)

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY, DELHI
OKHLA INDUSTRIAL ESTATE, PHASE III, NEAR GOVIND PURI METRO STATION, NEW DELHI,
DELHI-110020

I. Introduction

Network data is found in the form of metabolic networks, neural networks, etc. in biology, social networks, chemical networks and citation networks. In such networks, one can often find certain groups of nodes that are linked to one another more than the rest of the network. These groups of nodes may be termed as "Communities". Hence, community detection is essentially a graph partitioning problem.

Finding communities can be of utmost importance to recognize various features about the network. Communities in a protein interaction network correspond to cells with similar functionalities [1]. Communities in chemical networks may show the various sub-compounds that form a bigger more complex compound like a protein [2]. Communities in social networks and citations networks may show groups of individuals that interact with one another.

A "Community", however, is an abstract concept. There is no fixed definition of the same. It is in general accepted that nodes within a community have more number of edges within themselves compared to the rest of the network. Modularity [3] is a quality function which allows us to define a community, and is to date the most popular definition of a community in a network. Maximizing modularity for a particular network has been known to be one of the most successful blueprints for finding communities within a network.

Community detection, in general, however, is not an easy task. It is a classic combinatorial optimization problem. Maximizing Modularity in particular has been shown to be NP-Complete [4], APX-hard [6] and inapproximable upto any constant factor $\alpha > 0$ [5]. However, various algorithms following heuristics [7], [8] or proving certain additive guarantees [5], [9] have achieved state of the art results for a lot of networks of importance.

Still, networks can be of massive sizes, of the order of billions of nodes and edges. With various new discoveries and the increasing amount of data, this size is only going to get bigger and classical methods of computation would soon be found lacking. Quantum computational models have been known to provide asymptotic speedups over classical computational models and thus, provide an interesting avenue for further exploration of new community detection algorithms.

For this project, I have tried to formulate the modularity maximization problem as a novel Quadratic Unconstrained Binary Optimization (QUBO) problem. Quantum computers by D-Wave, such as the 2X and the 2000Q, have been used extensively in the recent past for solving problems of the QUBO form. Various works [10], [11], [12] formulate maximizing modularity as a QUBO problem and have demonstrated that this methodology can yield results very close, if not at par, to the state of the art. However, current quantum computers limit the size of the networks these algorithms can be run for. The biggest network any of the previous works have been able to use with their methods are only a few hundred nodes large. Though, it is the belief that with further engineering advancements, one would be able to leverage quantum computers for running these algorithms for larger networks and the need for algorithms that could leverage such systems in the future is very important.

The rest of this report is organized as follows. I define a few preliminaries in section 2, followed by an in depth review of the already existing models for community detection on quantum computers in section 3. I describe the work I did for this project in section 4. Finally, I conclude in section 5 along by describing a few possible directions in which I could take this project further.

II. Preliminaries

A. Modularity

Modularity is a quality function defined for a given partitioning, or, a community labelling, $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ of vertices in a graph G. It is defined as the fraction of edges that lie within these communities minus the fraction of edges that would have lied within these communities if the graph were random, where this random graph preserves the degree sequence of G. It is based on the idea that if the given community labelling $\mathcal{C}$ is good, more edges will lie within these communities than they would have if the graph were random. Based on this idea, any random labelling of communities should yield zero Modularity.

Modularity can be computed as follows:

$$Q(G, \mathcal{C}) = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{d(i)d(j)}{2m} \right] \delta^{\mathcal{C}}_{i,j} \tag{1}$$

where, A_{ij} is the adjacency matrix of G, $m = |E(G)|$ and $\delta^{\mathcal{C}}_{i,j}$ is the Kronecker delta function which yields 1 when v_i, v_j belong to the same cluster in $\mathcal{C}$ and 0 otherwise.

Here, we are ensuring that the term in the summation is non-zero only when 2 vertices are in the same cluster, where the first sub-term gives the number of edges lying within the communities in G and the second sub-term gives the number of edges that would have lied in G if it were a random graph as described above. We can alternately rewrite this function as:

$$Q(G, \mathcal{C}) = \sum_{C_i \in \mathcal{C}} \left[\frac{l_i}{m} - \left(\frac{d_i}{2m} \right)^2 \right], \tag{2}$$

where, $l_i = |E(G[C_i])|$ and d_i is the sum of the degrees of the vertices in C_i.

Further, Modularity, for a fixed number of communities k, can also be represented in the vector form as follows:

$$Q(G, \mathcal{C}) = \frac{1}{4} \mathbf{s}^T \mathbf{B} \mathbf{s}, \tag{3}$$

where, $\mathbf{s}$ is $n \times k$ binary matrix such that $s_{ij} = 1$ when vertex i is in community k. One should note that in general, the number of communities is not given, but has to be determined by finding the optimal partitioning.

The value of Modularity lies in [-1, 1) with a higher value indicating a better partitioning of the graph.

B. Quadratic Unconstrained Binary Optimization

Quadratic Unconstrained Binary Optimization or QUBO, is a mathematical formulation that allows us to model various combinatorial optimization problems. Any QUBO problem can be formulated as follows:

$$\arg\min_x \Big(\sum_i a_i x_i + \sum_{i<j} b_{i,j} x_i x_j \Big) \tag{4}$$

where, x is a n-length binary vector, $a \in \mathbb{R}^n$ and $b \in \mathbb{R}^{n \times n}$.

Quantum computers, such as the D-Wave 2000Q and 2X, act as Quantum annealers and can minimize an Ising objective Hamiltonian by finding its ground state energy using Adiabatic Quantum Computing. An Ising objective Hamiltonian $H(s)$ may be of the following form:

$$H(s) = \sum_i h_i s_i + \sum_{i<j} h_{i,j} s_{ij}, \tag{5}$$

where, $s_i \in \{-1, +1\}$ are magnetic spin variables where the result is encoded; h_i and J_{ij} are local magnetic fields and coupling strengths that encode the problem Hamiltonian.

One can simply put $s = 2x - 1$ to convert an Ising model to a QUBO model and hence, quantum computers can be used to solve QUBO problems. Although, it isn't necessary that one finds the true global minimum as minimizing QUBO problems is often NP-Hard. It is shown that quantum annealing can obtain a limited speedup over Classic simulated annealing [13] and thus, this provides an excellent framework for giving good algorithms for community detection.

However, the current quantum computers by D-Wave have physical constraints such as limited precision, sparse connectivity and limited available number of qubits. One can represent only 48 variables at a time on the 2X computer and only 64 variables on the 2000Q computer. Problems with larger number of variables need to broken down to smaller sub-problems and solved one by one on a quantum computer.

Modelling constrained optimization problems as QUBO problems [17]: QUBO problems are naturally unconstrained. Thus, it is slightly non-trivial how constrained optimization problems can be modelled as QUBO problems. One can do this, by adding penalty terms to the objective function for each constraint that admits the form $\sum a_i x_i + \sum b_{ij} x_i x_j$. These penalty terms are formulated such that violating these constraints can cause the objective function to incur a heavy penalty (reduction for maximization problems, blowing up for minimization problems).

For certain types of constraints, quadratic penalties for creating QUBO models are already known (See Table 1). Here, since the variables are binary, one can observe the undesired cases and create a conjunction or disjunction of the variables to form the penalty term. These penalty terms are simply added to the objective function. For example consider the Min-vertex Cut problem:

$$\begin{aligned} \min \quad & \sum_{i \in V} x_i \\ \text{s.t.} \quad & x_i + x_j \geq 1 \forall (i,j) \in E \\ & x_{ij} \in [0,1] \ \forall i, j \in V \end{aligned}$$

Constraint	Penalty
$x + y \leq 1$	$M(xy)$
$x + y \geq 1$	$M((1-x)(1-y))$
$x + y = 1$	$M((1-x)(1-y) + xy)$
$x \leq y$	$M(x(1-y)))$
$x + y + z \leq 1$	$M(xy + yz + xz)$
$x = y$	$M(x + y - 2xy)$

TABLE I: Table of a few known constraint/penalty pairs [17]

We can model the constraint as $M((1-x)(1-x_j))$. Hence, the QUBO formulation of the Min-vertex Cut problem as QUBO formulation is

$$\arg\min_x \sum_{i \in V} x_i + \sum_{(i,j) \in E} M(1-x)(1-x_j)$$

However, for constraints of the form $\sum a_i x_i = k$, where the summation consists of a lot of terms, forming such disjunctoins may be slightly difficult. In such cases, the penalty term has the general form $M(\sum a_i x_i - k)^2$. One can see that is the constraint is satisfied, the penalty is zero. For example, consider the exact Knapsack Problem:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} w_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} c_i x_i = B \\ & x_{ij} \in [0,1] \quad \forall i, j \in V \end{aligned}$$

We can model the constraint, as $M(\sum_{i=1}^{n} c_i x_i - B)^2$. Hence, the QUBO formulation for the exact Knapsack Problem is

$$\arg\min_x \sum_{i=1}^{n} w_i x_i + M(\sum_{i=1}^{n} c_i x_i - B)^2)$$

Modelling inequality constraints using the above form is however a bit tricky and expensive computationally. For example, now consider the general Knapsack Problem:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} w_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} c_i x_i \leq B \\ & x_{ij} \in [0,1] \quad \forall i, j \in V \end{aligned}$$

Here, we can't simply model the constraint as $M(\sum_{i=1}^{n} c_i x_i - B)^2$, because this adds a penalty for the cases which are allowed by the constraint. We can introduce extra binary slack variables that can help us to convert the equality. To do so, we would have to observe the minimum realizable value of the LHS. If say the minimum realizable value of the LHS is B'. Then, we need to add $\lceil \log(B - B') \rceil$ extra binary variables. The constraint can then be modelled as $M(\sum_{i=1}^{n} c_i x_i + \sum_{j=1}^{\lceil \log(B-B') \rceil} s_j - B)^2$. Thus, the QUBO formulation for the general Knapsack Problem is

$$\arg\min_{(x,s)} \sum_{i=1}^{n} w_i x_i + M(\sum_{i=1}^{n} c_i x_i + \sum_{j=1}^{\lceil \log(B-B') \rceil} s_j - B)^2)$$

III. Literature Review

The most successful community detection models follow 3 basic steps and only differ in the choice of how they perform the first 2 steps.

- They formulate the problem as a QUBO model.
- They divide a large problem into many subproblems so that it fits on a quantum device.
- They use the D-Wave Quantum Annealer framework to find communities on the sub-problems formed in the previous step.

Here I discuss 3 previous works [10], [11], [12] in depth that have followed this framework for community detection on quantum computers and further, a brief overview of a few important results about modularity maximization.

A. Network community detection on small quantum computers[11]

This work solves the 2-community detection problem. Since, they fix the number of communities to 2, they are able to write an unconstrained QUBO formulation for Modularity. Their formulation can also be used to solve problems on an IBM quantum computer using the VQE approach.

Since, they focus on a 2-community detection problem, they are able to get rid of the constraint and are able to drastically reduce the number of variables. They simply rewrite Modularity as follows:

$$Q = \sum_{i>j} 2\mathbf{B_{ij}s_i s_j} \tag{6}$$

Here, the variable $\mathbf{s_i}$ are simply spin states which when +1, indicate that vertex i is in community 1 and which when -1, indicate that vertex i is in community 2.

To fit the problem onto a quantun computer, they use a novel approach called the Quantum Local Search. It starts with a random assignment of vertices to communities. At each iteration, a few vertices that may bring the max possible increase in Modularity are chosen and put in a set X. Now, this subproblem is solved either on a D-Wave Quantum Annealer or an IBM Quantum Computer using VQE. This subproblem is solved while fixing the variables that are not a part of this subproblem. This procedure is repeated until convergence.

B. Multi-Community Detection in Signed Graphs Using Quantum Hardware [10]

This work is targeted for Community Detection in signed graphs, but it can be used for unsigned graphs as well. Their algorithm takes as input an upper bound on the number of communities that should be found in the graph. Their approach can be applied for any community detection metric that follows the QUBO model.

They use the matrix formulation of Modularity in (3) to form their QUBO model. Thus, their formulation has k variables for every vertex i. Further, they add a penalty term to incorporate the constraint, $|\mathbf{s_i}| = 1$ for every vertex i, where $\mathbf{s_i}$ is a v, to ensure that any vertex doesn't get assigned to multiple communities at a time. They write a penalized version of Modularity $Q^{\mathcal{P}}$ to form their QUBO model as follows:

$$Q^{\mathcal{P}} = \sum_{ij \in V} \mathbf{s_i B_{ij} s_j^T} - M \sum_{i \in V} (1 - |\mathbf{s_i}|)^2 \tag{7}$$

Here, the algorithm only takes in 2 hyperparameters, the value M and the upper bound on the number of communities k. Although, one might be tempted to set M too high such that violating the constraint would cause heavy penalties, it can be counterproductive as then the objective function increases very slowly. Regarding k, one can see that when the optimizer sees that a lesser number of communities would achieve a higher Modularity value, it will simply set $\mathbf{s}$ to 0 for the entire corresponding row. Again, one might be tempted to set k too high as it is an upperbound. But that increases the number of variables and reduces the speed of the optimizer. Here, the number of variables in the problem is $|V|k$.

The authors use a decomposition based Block-Coordinate Descent on top of each QUBO solver to solve the QUBO problems whose size exceeds the capacity of the quantum annealer. This approach iteratively solves such subproblems until covergence. See Figure 1 [10] for a more clear picture of this process.

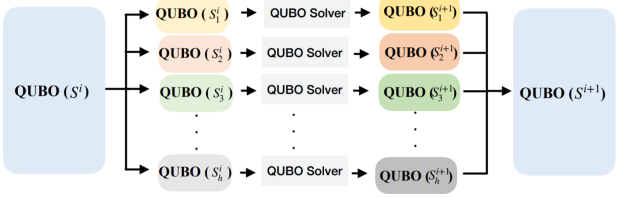

Fig. 1: Here a problem S^i is divided into subproblems $\{S_1^i, S_2^i, ..., S_h^i\}$. And each subproblem, S_j^i, is solved while keeping the variables not in S_j^i constant. Finally, the solutions to all subproblems S_j^{i+1} are combined to form the new solution S^{i+1} [10]

C. Detecting Multiple Communities Using Quantum Annealing on the D-Wave System [12]

This work follows a very similar path to the previous paper. They both came out independently at approximately the same time. Their QUBO formulation is the same as [10]. However, they don't specify their strategy to divide the larger problem into subproblems. They have simply called the D-Wave quantum annealer *qbsolv* [15] which takes the full problem in QUBO format and makes various calls to the D-Wave system to solve subQUBOs for global minimization followed by a classical tabu search for local minimization.

One should note that a comparison of the above algorithms is futile for now. This is because current technology only allows running these algorithms for very small graphs for which it is easier to attain maximum modularity.

D. Modularity Results

In this section, I briefly describe a few results about Modularity in literature that I leverage later to propose an alternate algorithm for community detection on quantum computers.

[16] proposed an Integer Linear formulation for Modularity as follows:

$$\begin{aligned} \max \quad & Q = \sum_{i,j \in V} B_{ij}(1 - x_{ij}) \\ \text{s.t.} \quad & x_{ij} + x_{jk} - x_{ik} \geq 0 \quad \forall i < j < k \\ & x_{ij} + x_{ik} - x_{jk} \geq 0 \quad \forall i < j < k \\ & x_{jk} + x_{ik} - x_{ij} \geq 0 \quad \forall i < j < k \\ & x_{ij} \in [0, 1] \quad \forall i, j \in V \end{aligned}$$

Here, $x_{ij} = 0$ if i, j are in the same communities. Further, the constraints essentially ensure that a metric relationship for x_{ij} is maintained. This ensures transitivity and consistency. However, the above equation has $O(n^3)$ constraints and is a bit slow to solve.

[8] proposed an alternate LP which is equivalent to the above IP, with a much lesser number of constraints.

$$\begin{aligned} \max \quad & Q = \sum_{i,j \in V} B_{ij}(1 - x_{ij}) \\ \text{s.t.} \quad & x_{ik} + x_{kj} \geq x_{ij} \quad k \in K(i,j) \setminus \{i,j\} \forall i < j \\ & x_{ij} \in [0, 1] \quad \forall i, j \in V \end{aligned} \tag{8}$$

where, $K(i, j)$ is the minimum vertex cut between the vertices, i, j. This bounds the number of constraints by $\max(\sum_i (i-1)d_i, m(n-1))$ where d_i is the degree sequence of the vertices arranged in non-increasing order.

IV. Work Done

I propose an alternative model for Community Detection on Quantum Computers using an alternative QUBO formulation based on the IP in (8). We can see that the objective function of this LP is already in the form discussed in (4). Now we incorporate the constraints of (8) into the QUBO formulation. As discussed above, to incorporate inequality constraints, one has to introduce binary slack variables to convert it into an equality constraint which we can later easily include in the QUBO in the form of a penalty.

If we observe any constraint, $x_{ik} + x_{kj} - x_{ij} \geq 0$, where $i, j \in V$ and $k \in K(i, j) \setminus \{i, j\}$, for binary values, the only forbidden case is when $x_{ik} = 0, x_{kj} = 0$ and $x_{ij} = 1$. Thus, we can add a penalty term $M((1 - x_{ik})(1 - x_{kj})x_{ij})$. Since, all variables are binary, this term is one only when the above case happens. Hence, for binary values, $x_{ik} + x_{kj} - x_{ij} \geq 0$, is equivalent to $((1 - x_{ik})(1 - x_{kj})x_{ij})$.

To incorporate it into the QUBO formualation, we add a term $M((1 - x_{ik})(1 - x_{kj})x_{ij})$ as described above. We can see that this preserves the form described in (4) (Note that since we have binary variables, we can simply convert any variable of the form x^2 to x).

Thus, the final QUBO formulation is as follows:

$$\begin{aligned} &\arg\max_x \textstyle\sum_{i,j \in V} B_{ij}(1 - x_{ij}) \\ &+ M \textstyle\sum_{i<j} \sum_{k \in K(i,j) \setminus \{i,j\}} M((1 - x_{ik})(1 - x_{kj})x_{ij}) \end{aligned}$$

The no. of variables in this formulation is $O(n^2)$. Now, I describe a procedure similar to the QLS approach in [11] to fit problems of large sizes onto quantum computers.

Initially, for all vertices, we assign communities uniformly at random and then obtain the resulting variables $x_{ij} \forall i, j \in V$. Once, we have an initial assignment of nodes, we fine tune it by selecting a random subgraph G' in G such that G' is connected. Such a subset can again be obtained using a Random/Quantum Walk. Then, we optimize Modularity by optimizing over all variables x_{ij} where i or $j \in G'$ and fixing all variables x_{ij} where i and j not $\in G'$. We continue this process until convergence. Note that such a selection of the subproblem is necessary. Unlike [11] which greedily choosed random vertices that may be far apart in the graph, we need a connected subgraph for this formulation as optimizing over randomly selected variables can lead to a violation of the constraints.

I now describe why I feel my algorithm might work better than the ones described in section 3 and describe a few reasons why it might be worse.

The biggest advantage of my algorithm is that it needs no prior assumption of k for solving the community detection problem. This might save a user valuable time.

However, the number of variables in the LP is too high which can cause the execution time to increase. In particular, the number of variables are higher by a factor of n/k. However, one can notice that there are a lot of variables that there are a lot of redundant variables in the LP. In particular, one can consider removing all variables that remain 0 for a lot of iterations. This redundancy is there because a few vertices might be very far away from each other in a graph. Although, one can't prove a bound on this and the performance is a bit suspect too.

There is another uncertainity in my algorithm regarding the selection of the subproblem. Modularity is known to be a global metric and solving subproblems locally might not be optimal for a metric of its nature. However, at the same time, selecting such subproblems can find near optimal solutions to such subproblems more quickly.

V. Future Directions

- I could not implement my algorithm on an actual quantum computer due to time constraints. Finding the results of this algorithm on an actual quantum computer can be very interesting considering the questions I have raised above.
- The variables in the QUBO problem are very high. Some way to reduce the number of variables, either by combining a few variables or ignoring variables can make the algorithm feasible and possibly better than the ones described previously. Doing this while keeping the constraints in mind makes this non-trivial.
- [5], [9] have proposed two separate SDP formulations of Modularity. [5] obtains it by reducing Modularity Maximization to the Max-Agree problem. Both have given additive approximation algorithms for Modularity Maximization using an SDP rounding approach. Quantum computers can also be leveraged to do this. There have been a lot of works that propose quantum algorithms for solving SDPs. However, these algorithms work very slowly for dense matrices like $\mathbf{B}$. Further advancements in this area can lead to an alternate unexplored direction for community detection.

- I also I would like to explore a more exact approach for community detection which gives some provable results. Right now, there is no traction about how accurately the problem may be solved (This is almost true for classical algorithms as well).

REFERENCES

[1] P. Ghale, M. Kroonblawd, S. Mniszewski, C. Negre, R. Pavel, S. Pino, V. Sardeshmukh, G. Shi, and G. Hahn. Task-based parallel computation of the density matrix in quantum-based molecular dynamics using graph partitioning. SIAM Journal on Scientific Computing, 39(6):C466–C480, 2017.

[2] J. Chen, B. Yuan, Detecting functional modules in the yeast protein–protein interaction network, Bioinformatics 2006.

[3] Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. Proceedings of the National Academy of Sciences, 99(12):7821–7826, 2002.

[4] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. Knowledge and Data Engineering, IEEE Transactions on, 20(2), 2008.

[5] T. N. Dinh, X. Li, and M. T. Thai. Network clustering via maximizing modularity: Approximation algorithms and theoretical limits. In ICDM '15: Proceedings of the 2015 IEEE International Conference on Data Mining, pages 101–110, 2015.

[6] Bhaskar DasGupta and Devendra Desai. On the complexity of newman's community finding approach for biological and social networks. Journal of Computer and System Sciences, (0), 2012.

[7] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10), 2008.

[8] T. N. Dinh and M. T. Thai, "Toward optimal community detection: From trees to general weighted networks," Internet Mathematics, vol. 11, no. 3, pp. 181–200, 2015.

[9] Y. Kawase, T. Matsui, and A. Miyauchi. Additive approximation algorithms for modularity maximization. arXiv preprint arXiv:1601.03316, 2016.

[10] E. Zahedinejad, D. Crawford, C. Adolphs, J. S. Oberoi, Multi-Community Detection in Signed Graphs Using Quantum Hardware, arXiv preprint arXiv:1901.04873 2019

[11] Shaydulin, R., Ushijima-Mwesigwa, H., Safro, I., Mniszewski, S. and Alexeev, Y. (2019), Network Community Detection on Small Quantum Computers. Adv. Quantum Technol., 2: 1900029.

[12] C. F. Negre, H. Ushijima-Mwesigwa, S. M. Mniszewski, Detecting Multiple Communities Using Quantum Annealing on the D-Wave System, arXiv preprint arXiv:1901.09756 2019.

[13] S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, M. Troyer (2014) Evidence for quantum annealing with more than one hundred qubits, Nature Physics, vol. 10, pp. 218-224.

[14] A. Lucas (2014) Ising Formulations of Many NP Problems, Frontiers in Physics, vol. 5, no. arXiv:1302.5843, p. 2.

[15] Mike Booth, Steven P. Reinhardt, and Aidan Roy (2017). Partitioning optimization problems for hybrid classical/quantum execution. D-Wave Technical Report Series, 14(1006A-A):1–9

[16] G. Agarwal and D. Kempe (2008). Modularity-maximizing graph communities via mathematical programming. Eur. Phys. J. B, 66.

[17] Fred Glover, Gary Kochenberger and Yu Du (2018), Quantum Bridge Analytics I: A Tutorial on Formulating and Using QUBO Models. arXiv:1811.11538.