QUBO formulation of some spanning tree related problems

Katalin Friedl*

Department of Computer Science and Information Theory
Budapest University of Technology and Economics
H-1111 Budapest, Műegyetem rakpart 3., Hungary
friedl@cs.bme.hu

László Kabódi*

Department of Computer Science and Information Theory
Budapest University of Technology and Economics
H-1111 Budapest, Műegyetem rakpart 3., Hungary
kabodil@cs.bme.hu

Abstract: The appearance of quantum annealers, like the D-Wave machines, made it interesting to formulate problems as Quadratic Unconstrained Binary Optimization tasks. In this paper we show two new QUBO formulations to the spanning tree problem, and use it to find a minimum weight spanning tree, a minimum weight spanning tree with a maximal degree constraint and a Steiner tree. We show that variants of the classical Prüfer sequences can be used for this purpose and they are as good as earlier results. Our network flow based approach improves on the earlier results when the number of edges is at most $O(\frac{n^2}{\log n})$.

Keywords: QUBO, spanning tree, Steiner tree

1 Introduction

Quantum computers promise significant speed ups in some problem classes. One of these classes is Quadratic Unconstrained Binary Optimization (QUBO), which is implemented, for example, in D-Wave quantum annealers. An interesting area of research is to create QUBO formulations for well-known classical problems. In this paper we show two new QUBO formulations to the spanning tree problem, and use them to find a minimum weight spanning tree, a minimum weight spanning tree with a maximal degree constraint and a Steiner tree. In Section 2 we give a brief introduction to QUBO, and show some basic tools that can be used when formulating a task in QUBO. In Section 3 we give a formulation for finding an arbitrary spanning tree in a given graph. Section 4 shows how to extend this to find a minimum weight spanning tree. Section 5 further extends this to a bounded degree minimum weight spanning tree. In Section 6 we show how to use the same approach to find an arbitrary Steiner tree. Section 7 contains our conclusions.

2 QUBO

A quantum annealer, like the D-Wave machines, finds the minimum of a given quadratic function over binary variables. In quantum computing, the function to be minimized is also called the Hamiltonian, and the binary variables are called qubits.

Formally, a QUBO task is to find the minimum of a function $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \alpha_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{i,j} x_i x_j$ over the binary variables $x_1, \ldots, x_n$. Creating a QUBO formulation for a problem means mapping binary variables to the parameters of the task, and giving a function, the minimum of which is where the value of the variables gives the solution of the original task.

2.1 Example: unary encoding

A number $1 \leq k \leq n$ can be encoded into n qubits $x_1, \ldots, x_n$ in such a way, that $x_k = 1$ and $x_i = 0$ for all $i \neq k$. The Hamiltonian for this encoding is $H(x_1, \ldots, x_n) = (1 - \sum_{i=1}^{n} x_i)^2$. The minimum value of the

*This research was partially supported by the Ministry of Culture and Innovation and the National Research, Development and Innovation Office within the Quantum Information National Laboratory of Hungary (Grant No. 2022-2.1.1-NL-2022-00004).

Back to Contents

function is 0, and at all of the minimum points there is exactly one i such that $x_i = 1$.

2.2 Example: permutation

Another possible task is to place rooks on a chess board, one in each row and column, also known as a permutation. For this, qubits $x_{i,j}$ can be used, where $x_{i,j} = 1$ means that there is a rook on the ith row of the jth column. The Hamiltonian should have its minimum value where

(1) for each i there is exactly one j such that $x_{i,j} = 1$, and

(2) for each j there is exactly one i such that $x_{i,j} = 1$.

A Hamiltonian for **(1)** is $H_1 = \sum_{i=1}^{n}(1 - \sum_{j=1}^{n} x_{i,j})^2$. Similarly, for **(2)** is $H_2 = \sum_{j=1}^{n}(1 - \sum_{i=1}^{n} x_{i,j})^2$. The value of both Hamiltonians is 0 if and only if their respective constraint is true. So, they can be added together to get a Hamiltonian for the original problem: $H_{rook} = H_1 + H_2$. In H_{rook} only those instances have 0 value, where both $H_1 = 0$ and $H_2 = 0$, so exactly the solutions for the rook problem.

2.3 Example: binary encoding

Another frequent task is to have at most k out of n qubits $x_1, \ldots, x_n$ with value 1. Introduce k new qubits $y_1, \ldots, y_k$, where $y_i = 1$ if and only if the number of 1 valued x qubits is exactly i. The Hamiltonian H_y makes sure that exactly one y qubit have the value 1: $H_y = (1 - \sum_{i=0}^{k} y_i)^2$. Using the y qubits Hamiltonians can be written to constraint the number of 1 valued x qubits: $H_k = (\sum_{i=1}^{k} i \cdot y_i - \sum_{j=1}^{n} x_j)^2$. Adding these two Hamiltonians together, the task is solved. The Hamiltonian H_y makes sure that there is exactly one ℓ such that $y_\ell = 1$, so in H_k the first sum evaluates to ℓ. With that, $H_k = 0$ if and only if the number of x qubits that have value 1 is exactly $\ell \leq k$.

This solution works, but uses k additional qubits, and the number of qubits in a quantum computer is low. So, it is interesting to see how this can be done with less additional qubits. Previously the encoding of the number of 1s were unary, but binary encoding can also be used with only $\log k$ qubits. Let k be a power of 2. Then the Hamiltonian is $H_k = (\sum_{i=0}^{\log k} 2^i \cdot y_i - \sum_{j=1}^{n} x_j)^2$. Here the first sum evaluates to a number between 1 and k, which constraints the number of 1 valued x qubits to that number. Another benefit of binary encoding is that a Hamiltonian corresponding to H_y is not needed. If k is not a power of 2, the idea still can be used, with a bit more complicated Hamiltonian. Let $m = \lfloor \log k \rfloor$, then $H_k = \left(\sum_{i=0}^{m-1} 2^i \cdot y_i + (k + 1 - 2^m) - \sum_{j=1}^{n} x_j \right)^2$ works.

3 Spanning tree

Given an undirected graph $G = (V, E)$, let $V = \{1, 2, \ldots, n\}$ be the set of vertices and E the set of edges. Let $|E| = e$. Also let E' be the set of orienting edges that we obtain from E by directing every edge in both directions. The task is to get a spanning tree of G, a tree which contains all vertices of G.

3.1 Previous results

In [4] Lucas gave a formulation using $O(n \cdot e)$ qubits. The idea behind this formulation is that every tree can be considered as a rooted, leveled tree, with only the root at the top level. In this tree for each non-root vertex there is exactly one edge connecting it to higher levels, and that edge connects it with the level directly above it. This formulation sets these constraints using qubits for the edges and the vertices.

In his Master's thesis [2] Fowler used a different idea with only $O(n^2)$ qubits. The main idea is that a spanning tree (with some orientation) has a topological order. This formulation uses constraints to get the topological order using qubits for vertex pairs.

In [1] Carvalho does something similar. First, he creates a permutation of the vertices. Then he uses similar constraints as Fowler to get a spanning tree. In this formulation qubits are used for vertices. This method also uses $O(n^2)$ number of qubits.

Back to Contents

3.2 Our new formulations

We present two new formulations. One based on network flows, the other based on Prüfer sequences.

3.2.1 Formulation based on a network flows

The spanning tree problem can be formulated as a maximum flow in a network. Let vertex 1 be the source, that will be the root of the spanning tree. We replace each edge with two directed edges all with capacity $n-1$. Then add a new $(n+1)$st vertex, that is the sink, and new edges from vertices $2,\ldots,n$ to the sink, all with capacity 1. In this graph, maximal flows correspond to spanning trees in the original graph. Furthermore, every spanning tree corresponds to a maximal flow. To describe maximal flows, we use a method already appeared in [3]. The usual constraints are that for every vertex the size of the incoming flow has to be the same as the size of the outgoing, except for the source and the sink. The size of the outgoing flow from the source has to be the same as the size of the incoming flow to the sink, that has value $n-1$. To prevent a maximal flow to use cycles, we limit the edges with non-zero flow values to $n-1$, not counting the edges that go to the sink.

The flow on the edges is described using unary encoding. Then there are n qubits for every edge $(u,v)\in E'$: $x_{u,v,i}$, where $i=0,\ldots,n-1$ is the possible value of the flow on the directed edge (u,v). To formulate the Hamiltonian for this problem, we need the following:

(1) For every edge, there is a (maybe zero valued) flow:

$$H_f = \sum_{\substack{(u,v)\in E'\\ v\neq n+1}} \left(1-\sum_{i=0}^{n-1} x_{u,v,i}\right)^2 + \sum_{(u,(n+1))\in E'} \left(1-\sum_{i=0}^{1} x_{u,(n+1),i}\right)^2$$

(2) The flow conservation rule must hold for all vertices except the source and the sink:

$$H_c = \sum_{u=2}^{n} \left(\sum_{(v,u)\in E'} \sum_{i=0}^{n-1} i\cdot x_{v,u,i} - \sum_{(u,w)\in E'} \sum_{i=0}^{n-1} i\cdot x_{u,w,i} \right)^2$$

(3) The sum of the outgoing flow from the source is $n-1$:

$$H_{so} = \left((n-1) - \left(\sum_{(1,u)\in E'} \sum_{i=0}^{n-1} i\cdot x_{1,u,i} \right) \right)^2$$

(4) The sum of the incoming flow to the sink is $n-1$:

$$H_{si} = \left((n-1) - \left(\sum_{(u,(n+1))\in E'} \sum_{i=0}^{n-1} i\cdot x_{u,(n+1),i} \right) \right)^2$$

(5) There are exactly $n-1$ edges that have greater than 0 flow:

$$H_e = \left((n-1) - \left(\sum_{\substack{u,v\in E'\\ v\neq n+1}} \sum_{i=1}^{n-1} x_{u,v,i} \right) \right)^2$$

Adding these together the Hamiltonian is $H_{ufl} = H_f + H_c + H_{so} + H_{si} + H_e$. The value of H_{ufl} is zero, if the edges that have greater than 0 flow form a spanning tree in the graph, and greater than zero otherwise. The number of used qubits is $O(n\cdot e)$.

Improving on this, the flow values can be encoded in binary. Then we only need $2\cdot\log n$ qubits for each edge. Assume that n is a power of 2. If it is not, we can use the smallest power of 2 that is greater than n, the results will be the same. In this encoding $x_{u,v,0},\ldots,x_{u,v,\log n}$ are the bits of the binary value of the flow going through the edge (u,v). We also need some extra qubits to count the number of edges that have greater than zero flow, $y_{u,v,i}$: there are exactly i ones in the binary form of the flow value on edge (u,v). The constraints for the y qubits will be in **(6')** and **(7')**.

Back to Contents

In the case of binary encoding, there is no need for a Hamiltonian corresponding to H_f.

(2') The flow conservation rule must hold for all vertices except the source and the sink:

$$H_c = \sum_{u=1}^{n} \left(\sum_{(v,u)\in E'} \sum_{i=0}^{\log n} 2^i \cdot x_{v,u,i} - \sum_{(u,w)\in E'} \sum_{i=0}^{\log n} 2^i \cdot x_{u,w,i} \right)^2$$

(3') The sum of the outgoing flow from the source is $n-1$:

$$H_{so} = \left((n-1) - \left(\sum_{(0,u)\in E'} \sum_{i=0}^{\log n} 2^i \cdot x_{0,u,i} \right) \right)^2$$

(4') The sum of the incoming flow to the sink is $n-1$:

$$H_{si} = \left((n-1) - \left(\sum_{(u,(n+1))\in E'} \sum_{i=0}^{\log n} 2^i \cdot x_{u,(n+1),i} \right) \right)^2$$

(5') There are $n-1$ edges that have greater than 0 flow:

$$H_e = \left((n-1) - \left(\sum_{\substack{(u,v)\in E' \\ v\neq n+1}} \sum_{i=1}^{\log n} y_{u,v,i} \right) \right)^2$$

(6') For every edge exactly one of $y_{u,v,i}$ qubits are 1, the rest are 0:

$$H_{y1} = \sum_{(u,v)\in E'} \left(1 - \sum_{i=0}^{\log n} y_{u,v,i} \right)^2$$

(7') The y qubits really count the number of ones in the x qubits:

$$H_{y2} = \sum_{(u,v)\in E'} \left(\sum_{i=0}^{\log n} i \cdot y_{u,v,i} - \sum_{j=0}^{\log n} x_{u,v,j} \right)^2$$

The Hamiltonian is $H_{bfl} = H_c + H_{so} + H_{si} + H_e + H_{y1} + H_{y2}$. Similarly to the previous one, $H_{bfl} = 0$ if and only if the edges that have greater then 0 flow form a spanning tree, otherwise $H_{bfl} > 0$. This method uses $O(e \log n)$ qubits.

3.2.2 Formulation based on Prüfer sequence

A Prüfer sequence is a way to encode a tree in $(n-2)$ numbers. Recall that to create a Prüfer sequence from a tree, first number the vertices in an arbitrary order. Then find the leaf with the smallest number, remove it from the graph, and write down the number of its neighbour. Repeat this until there are only two vertices left in the tree. Given a Prüfer sequence, the tree can be reconstructed. Add n as the $(n-1)$st element to the sequence. Then find the smallest number that does not appear in the sequence. That was the first leaf deleted, so connect it to the first element of the sequence. The ith deleted vertex is the smallest one not appearing in the sequence after the ith element, and not reconstructed earlier.

Based on this, we can create a spanning tree for a given graph by creating a Prüfer sequence that decodes to a spanning tree of the graph. For this purpose we need n qubits for each of the $(n-2)$ elements of the Prüfer sequence: $x_{i,j}$ is 1 if the ith element in the Prüfer sequence is j. And $(n-1)$ qubits for each of the $(n-1)$ reconstructed vertices: $y_{i,j}$ is 1 if the ith reconstructed vertex is j. We need the following Hamiltonians:

Back to Contents

(1) Every element of the sequence takes only one value:

$$H_x = \sum_{i=1}^{n-2} \left(1 - \sum_{j=1}^{n} x_{i,j} \right)^2$$

(2) Every reconstructed element takes only one value:

$$H_{y1} = \sum_{i=1}^{n-1} \left(1 - \sum_{j=1}^{n-1} y_{i,j} \right)^2$$

(3) Every vertex (except the nth one) was reconstructed exactly once:

$$H_{y2} = \sum_{j=1}^{n-1} \left(1 - \sum_{i=1}^{n-1} y_{i,j} \right)^2$$

(4) Every reconstructed vertex is such, that it was not reconstructed before and it does not appear later in the sequence:

$$H_{d1} = \sum_{i=1}^{n-1} \left(\sum_{j=1}^{n-1} y_{i,j} \left(\sum_{k=1}^{i-1} y_{k,j} + \sum_{l=i}^{n-2} x_{l,j} \right) \right)$$

(5) The edges implied by the Prüfer sequence really appear in the graph:

$$H_e = \sum_{i=1}^{n-2} \sum_{(u,v) \notin E'} y_{i,u} x_{i,v} + \sum_{(u,n) \notin E'} y_{(n-1),u}$$

(6) The reconstructed vertex has the smallest value out of the possible vertices:

$$H_{d2} = \sum_{i=1}^{n-1} \sum_{j=1}^{n-1} (n-i)\,(j-1)\, y_{i,j}$$

In **(4)** for each i if the ith reconstructed vertex is j, the second part must be zero. So the vertex j can't be reconstructed before the ith element, and it can't appear later in the sequence. If the ith reconstructed vertex is not j, then the second part can be anything, the product will be zero.

In **(5)** we penalize if there is an edge implied by the sequence that does not appear in the graph.

In **(6)** each y qubit get a unique weight in such a way, that enforces that if there are multiple possible vertices to reconstruct, the smallest one will be chosen.

For this to work, the other constraints have to be quite strong. The maximum value of $H_{d2} < 0 + n + 2n + \ldots (n-1) \cdot n < n^3$.

So, the final Hamiltonian is $H_{pr} = n^3(H_x + H_{y1} + H_{y2} + H_{d1} + H_e) + H_{d2}$, which has a value greater than n^3 if the result is not a spanning tree, and less than n^3 if it is one. This method uses $O(n^2)$ qubits.

4 Minimum weight spanning tree

In the previous section, we searched for any spanning tree. But if the graph has weights on its edges, the goal usually is to find a minimum weight spanning tree. The weight of a tree is defined as the sum of the weights on the edges of the tree. Let W denote the sum of the weights in the graph.

If the Hamiltonian H is such, that it takes its global minimum if and only if the represented graph is a spanning tree of G, then the Hamiltonian $H' = W \cdot H + H_w$, where H_w is the weight of the spanning tree given by H.

In their solutions Lucas and Fowler have qubits for edges, in this case H_w is easy to express.

Carvalho does not have qubits for the edges, but the edges can be calculated without additional qubits.

As they did not use any additional qubits, the number of qubits remains $O(n \cdot e)$, $O(n^2)$ and $O(n^2)$, respectively.

Back to Contents

4.1 Formulation based on a network flow

In the network flow based formulation, we also have all the necessary qubits to get the edges that are in the spanning tree.

In the unary case the weight of the spanning tree is $H_w = \sum_{(u,v)\in E'} w_{u,v} \sum_i x_{u,v,i}$. Given this, the Hamiltonian for the whole task is

$$H_{uflmin} = H_w + W \cdot (H_f + H_c + H_{so} + H_{si} + H_e).$$

The binary case is quite similar, the difference is that we use the y qubits here: $H_w = \sum_{(u,v)\in E'} w_{u,v} \sum_{i=1}^{\log(n)} y_{u,v,i}$, so the final Hamiltonian is

$$H_{bflmin} = H_w + W \cdot (H_c + H_{so} + H_{si} + H_e + H_{y1} + H_{y2}).$$

As we did not use any additional qubits, the number of qubits remains $O(n \cdot e)$ in the unary case and $O(e \log n)$ in the binary case.

4.2 Formulation based on Prüfer sequence

In this formulation, adding the weight to the Hamiltonian won't work, because the different spanning trees have different values. Without **(6)**, $H'_{pr} = H_x + H_{y1} + H_{y2} + H_{d1} + H_e$ is zero if and only if the selected edges form a spanning tree. In this formulation a Prüfer sequence does not uniquely represent a spanning tree, but the main thing is, that the decoded graph is still a spanning tree. Because the first deleted vertex (does not matter if it is the smallest or not) is a leaf, as it is required in a Prüfer sequence. So, it does not appear in any cycle. Removing that vertex from the graph, the second deleted vertex is also a leaf in the remaining graph, so it also cannot be in a cycle, etc. And because the graph has $(n-1)$ edges, it must be a spanning tree on n vertices.

The weight Hamiltonian: $H_w = \sum_{i=1}^{n-1} \sum_{(u,v)\in E'} w_{u,v} y_{i,u} x_{i,v} + \sum_{(u,n)\in E'} w_{u,n} y_{(n-1),u}$. The full Hamiltonian is

$$H_{prmin} = H_w + W \cdot (H_x + H_{y1} + H_{y2} + H_{d1} + H_e).$$

As we did not use any additional qubits, the number of qubits remains $O(n^2)$.

5 Minimum weight spanning tree with a maximal degree constraint

Although the minimum weight spanning tree problem is an algorithmically easy task, the previously presented methods are useful for more difficult problems. One of such tasks is the minimum weight spanning tree with a maximal degree constraint. Here we consider that there is an additional constraint for the maximum degree in the spanning tree. This makes the task NP-complete, since it contains the Hamiltonian path problem, and there is no known classical algorithm to efficiently solve it.

Let this maximum degree be Δ. Lucas, Fowler and Carvalho all do the same for this task. For all vertices, they introduce Δ new qubits. With those, they constraint the degree of each vertex to be between 1 and Δ. This needs $O(\Delta \cdot n)$ additional qubits, which keeps the number of qubits as $O(n \cdot e)$, $O(n^2)$ and $O(n^2)$ respectively. By using binary encoding, the number of additional qubits can be reduced to $O(\log \Delta \cdot n)$.

5.1 Formulation based on a network flow

The same approach can be used here. For unary encoding of network flows, we use unary encoding for the degree constraint. Let $z_{i,k} = 1$ if and only if the degree of the ith vertex is k. With this the constraint can be written as:

$$H_\Delta = \sum_{u=1}^{n} \left(\sum_{k=0}^{\Delta} k \cdot z_{u,k} - \sum_{(u,v)\in E'} \sum_{i=1}^{n-1} x_{u,v,i} \right)^2.$$

Back to Contents

Because of the unary encoding, an additional Hamiltonian is needed: $H_z = \sum_{u=1}^{n}(1 - \sum_{k=1}^{\Delta} z_{u,k})^2$. Both of these Hamiltonians have 0 as their minimum value, so the final Hamiltonian is:

$$H_{uflmindeg} = H_w + W \cdot (H_f + H_c + H_{so} + H_{si} + H_e + H_z + H_\Delta).$$

The number of qubits needed is still $O(n \cdot e)$.

The binary case is quite similar, but with binary encoding for the degree constraint, provided Δ and n are powers of 2:

$$H_\Delta = \sum_{u=1}^{n}\left(\sum_{k=0}^{\log\Delta} 2^k \cdot z_{u,k} - \sum_{(u,v)\in E'}\sum_{i=1}^{\log n} y_{u,v,i}\right)^2.$$

For a fixed vertex u the first sum is its degree. The second sum goes through all the edges incident to vertex u in the original graph, and sums up the y qubits for flow values at least 1. So the value of the second sum is the number of edged incident to u in the spanning tree. Binary encoding does not need a Hamiltonian corresponding to H_z, so the final Hamiltonian in this case is:

$$H_{bflmindeg} = H_w + W \cdot (H_c + H_{so} + H_{si} + H_e + H_{y1} + H_{y2} + H_\Delta).$$

As with the rest of the solutions, the magnitude of the needed qubits does not change: $O(e \log n)$

5.2 Formulation based on Prüfer sequence

Here, a property of the Prüfer sequence can be used. In a Prüfer sequence, the degree of every vertex is the number they appear in the code plus 1. The encoding of the degree can be either unary or binary. Here, we provide the unary version, because the solution without the degree constraint already uses $O(n^2)$ qubits, but the binary version can be used without any additional thought. The Hamiltonian for the degree constraint is:

$$H_\Delta = \sum_{u=1}^{n}\sum_{k=1}^{\Delta}\left((k \cdot z_{u,k} - 1) - \sum_{i=1}^{n-2} x_{i,u}\right)^2.$$

The final Hamiltonian includes H_z as well, because the unary encoding:

$$H_{prmindeg} = H_w + W \cdot (H_x + H_{y1} + H_{y2} + H_{d1} + H_e + H_z + H_\Delta).$$

The number of qubits used remains $O(n^2)$.

6 Steiner tree

Another NP-hard task is to find a Steiner tree in a graph. Given a set of vertices $S \subseteq V$, the so-called Steiner vertices, a Steiner tree is a minimum weight subtree of the graph that contains all of the vertices in S, and may contain some other vertices as well.

In the solution of Lucas, a plus qubit is needed for each non-Steiner vertex, to encode if that vertex is in the Steiner tree or not. Using this, the other constraints can be limited to only the vertices in the Steiner tree. The number of qubits is still $O(n \cdot e)$.

The solution by Fowler does not require any additional qubits. He modifies the constraints, so for a vertex that is not in S, it is not needed, to have an edge going to it. So the number of qubits is still $O(n^2)$

Carvalho does not give a solution for the Steiner tree problem.

6.1 Formulation based on a network flow

With both the unary and binary encoding, the formula does not change, only the network has to be built a different way. The edges to the sink are needed only from the Steiner vertices, and the value of the flow is at most $|S|$.

This changes the number of used qubits to $O(e \cdot |S|)$ and $O(e \log |S|)$ for the unary and binary encoding respectively.

Back to Contents

6.2 Formulation based on Prüfer sequence

To create a Steiner tree, not all vertices are needed, but the number of vertices needed is not known beforehand. This solution still uses $n-2$ qubits for the Prüfer sequence. The idea is to allow the sequence to start with a few 0s, effectively shortening the sequence. To make sure, that the encoded graph is connected, every vertex in the sequence has to appear as a reconstructed one. Because the vertex n is always in the tree, one of the Steiner vertices should be n. As with the minimum weight spanning tree problem, we do not want a unique reconstruction for a given sequence, so **(6)** is not needed. The Hamiltonians for this problem:

(1) Every element of the sequence takes only one value:

$$H_x = \sum_{i=1}^{n-2} \left(1 - \sum_{j=1}^{n} x_{i,j} \right)^2$$

(2) Every reconstructed element takes only one value:

$$H_{y1} = \sum_{i=1}^{n-1} \left(1 - \sum_{j=1}^{n-1} y_{i,j} \right)^2$$

(3') Every Steiner vertex (except the nth one) was reconstructed exactly once:

$$H_{y2} = \sum_{j \in S} \left(1 - \sum_{i=1}^{n-1} y_{i,j} \right)^2$$

(4) Every reconstructed vertex is such, that it was not reconstructed before and it does not appear later in the sequence:

$$H_{d1} = \sum_{i=1}^{n-1} \left(\sum_{j=1}^{n-1} y_{i,j} \left(\sum_{k=1}^{i-1} y_{k,j} + \sum_{l=i}^{n-2} x_{l,j} \right) \right)$$

(5) The edges implied by the Prüfer sequence really appear in the graph:

$$H_e = \sum_{i=1}^{n-2} \sum_{(u,v) \notin E'} y_{i,u} x_{i,v} + \sum_{(u,n) \notin E'} y_{(n-1),u}$$

(7) If an element in the Prüfer sequence is 0, then the element before that must also be 0:

$$\sum_{i=2}^{n-2} \left(x_{i,0} \cdot \sum_{j=1}^{n} x_{i-1,j} \right)$$

(8) If a reconstructed element is 0, then the previously reconstructed element is 0 as well:

$$\sum_{i=2}^{n-1} \left(y_{i,0} \cdot \sum_{j=1}^{n} y_{i-1,j} \right)$$

(9) The ith element of the sequence is zero if and only if the ith reconstructed element is zero:

$$\sum_{i=1}^{n-2} \left(x_{i,0} \cdot \sum_{j=1}^{n} y_{i,j} \right) + \sum_{i=1}^{n-2} \left(y_{i,0} \cdot \sum_{j=1}^{n} x_{i,j} \right)$$

(10) Every element in the sequence must appear as a reconstructed vertex as well:

$$\sum_{j=1}^{n} \sum_{i=1}^{n-2} \left(x_{i,j} \cdot (1 - \sum_{k=1}^{n-2} y_{k,j}) \right)$$

For **(10)** notice, that $\sum_{k=1}^{n-2} y_{k,j}$ is either 0 or 1, depending on vertex j appearing in the tree.

There is one additional qubit for each element in the sequence and one for each reconstructed vertex. So the overall number of qubits needed is still $O(n^2)$.

Back to Contents

	Number of qubits	Steiner tree
Lucas [4]	$O(n \cdot e)$	yes
Fowler [2]	$O(n^2)$	yes
Carvalho [1]	$O(n^2)$	no
Unary flow	$O(n \cdot e)$	yes
Binary flow	$O(e \log n)$	yes
Prüfer sequence	$O(n^2)$	yes

Table 1: Qubits used by the different formulations. The last three are from this paper.

7 Conclusion

We gave two new formulations for spanning trees, minimum weight spanning trees, minimum weight spanning trees with a degree constraint and the Steiner tree problem, based on network flow and Prüfer sequence. The number of qubits used by previous methods compared to ours can be seen in Table 1. The second column shows the number of qubits used in the formulation, and the third shows if there is a formulation for the Steiner tree problem. There is only one column for the number of qubits, because the order of qubits used is the same for all examined tasks.

The unary encoded network flow formulation does not have any advantage over the binary encoded one, so it is only useful for explanatory purposes, as it can be easier to understand. The formulation by Lucas always uses the greatest number of qubits, but based on the number of the edges in the graph, the binary encoded flow can be more efficient than Fowler's and Carvalho's. If the number of the edges is less, then $O(\frac{n^2}{\log n})$, then the network flow based solution uses less qubits, but with a graph with $O(n^2)$ edges, the other two is more efficient.

References

[1] Ivan Carvalho. QUBO formulations for NP-hard spanning tree problems. *arXiv preprint arXiv:2209.05024*, 2022.

[2] Alexander Fowler. Improved QUBO formulations for D-Wave quantum computing. Master's thesis, University of Auckland, 2017.

[3] Thomas Krauss, Joey McCollum, Chapman Pendery, Sierra Litwin, and Alan J Michaels. Solving the max-flow problem on a quantum annealing computer. *IEEE Transactions on Quantum Engineering*, 1:1–10, 2020.

[4] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in physics*, 2:5, 2014.