A QUBO Formulation for Minimum Loss Spanning Tree Reconfiguration Problems in Electric Power Networks

Filipe F. C. Silva\textsuperscript{1,2,3}, Pedro M. S. Carvalho\textsuperscript{1,3}, Luís A. F. M. Ferreira\textsuperscript{1,3}, and Yasser Omar\textsuperscript{2,3,4}

\textsuperscript{1}INESC-ID, Sustainable Power Systems Group, Portugal
\textsuperscript{2}Instituto de Telecomunicações, Physics of Information and Quantum Technologies Group, Portugal
\textsuperscript{3}Instituto Superior Técnico, University of Lisbon, Portugal
\textsuperscript{4}Portuguese Quantum Institute, Portugal

March 2022

Abstract

We introduce a novel quadratic unconstrained binary optimization (QUBO) formulation for a classical problem in electrical engineering – the optimal reconfiguration of distribution grids. For a given graph representing the grid infrastructure and known nodal loads, the problem consists in finding the spanning tree that minimizes the total link ohmic losses. A set of constraints is initially defined to impose topologically valid solutions. These constraints are then converted to a QUBO model as penalty terms. The electrical losses terms are finally added to the model as the objective function to minimize. In order to maximize the performance of solution searching with classical solvers, with hybrid quantum-classical solvers and with quantum annealers, our QUBO formulation has the goal of being very efficient in terms of variables usage. A standard 33-node test network is used as an illustrative example of our general formulation. Model metrics for this example are presented and discussed. Finally, the optimal solution for this example was obtained and validated through comparison with the optimal solution from an independent method.

1 Introduction

Electrical distribution grids are facing the fundamental paradigm shift of becoming low-carbon smart grids. Under this scenario, the grids must be able to accept an increasing amount of distributed renewable energy generation (e.g., from urban and residential photovoltaic panels and small wind turbines) and of electric vehicles load. Handling the volatile nature of new load and generation poses new challenges to the grid operation, since the network must be regularly reconfigured in order to operate with minimum energy losses [1]. Those reconfigurations consist of switching on and off several network links. As the network reacts to the constant changes on load and generation profiles, the reconfiguration decisions need to be fast and optimal in order to be effective.

In this paper, the distribution grid is represented through a network graph where one of the nodes is the substation and the remaining nodes are the network loads. The network links are the electrical lines connecting the nodes. Figure 1 represents the 33-node test network used in this
Figure 1: The standard 33-node test network. The substation is node 0.
QUBO model is the sum of all constraint expressions. An optimal solution for this model complies with all CSP constraints, thus being a valid solution.

4. Add electrical losses terms to the QUBO model. The optimal solution for the complete QUBO model is the one minimizing electrical losses while still being topologically valid.

This paper is organized as follows. After this Introduction, the optimization problem is described in Section 2. Afterwards, in Section 3, the problem constraints are formulated and, in Section 4, the method of obtaining the QUBO model is described. Then, in Section 5, metrics and validation of the QUBO model are presented and discussed. Finally, Section 6 concludes the paper.

Throughout the paper, the terms network, node and link are used when node and link intrinsic attributes are important in the context (such as node load and link resistance in an electrical network), and the terms graph, vertex, edge and arc are used when the focus is on graph theory or topology.

2 Problem Description

The model of an electrical network includes the connectivity graph \( G = (V, E) \), the complex phasor of nodal electrical load current\(^1\) \( I_L^n \) and link electrical resistance\(^2\) \( R_{uv} \), with \( n \in V \) and \( (u, v) \in E \). Given this model as the input, the problem goal is to find the spanning tree \( T^* \) of \( G \) which minimizes the sum of active power losses on each link of the tree, as defined by

\[
T^* = \arg \min_{T \in \text{ST}(G)} \sum_{(u,v) \in E(T)} L_{uv}(T)
\]

where \( \text{ST}(G) \) is the set of all spanning trees of \( G \), \( E(T) \) is the set of all links on \( T \) and \( L_{uv}(T) \) is the power losses function on link \( (u, v) \) with tree \( T \). This function is given by

\[
L_{uv}(T) = R_{uv} |I_{uv}(T)|^2
\]

where \( I_{uv}(T) \) is the complex phasor of the electrical current flowing on link \( (u, v) \) with tree \( T \). This current is given by

\[
I_{uv}(T) = \sum_{n \in D_{uv}(T, v_0)} I_L^n
\]

where \( D_{uv}(T, v_0) \) is the set of all nodes downward of link \( (u, v) \) across \( T \), i.e., the nodes with a path to the root node \( v_0 \in V \) (the substation) along \( T \) which includes the link \( (u, v) \). This set includes either \( u \) or \( v \) depending on \( T \). \( D_{uv}(T, v_0) \) can be extended for the case \( (u, v) \notin E(T) \) yielding the empty set.

2.1 Problem partitioning into biconnected components

Let \( D_{m}(T, v_0) \) be the set of all nodes downward of node \( m \) across \( T \). If, for a given tree \( T \), \( m \) is itself downward of a link \( (u, v) \), then a node \( n \) downward of \( m \) is also downward of \( (u, v) \), i.e.,

---

\(^1\)Electrical loads are typically modeled as constant power (PQ loads), but we are modeling them as constant current loads since this allows a significant simplification to our losses model without changing the optimal configuration for the 33-node network. We are also assuming that voltage angle is zero across the network.

\(^2\)Since load current does not depend on load voltage and we are only concerned with active power losses, longitudinal link reactance can be ignored. Also, the 33-node network model does not specify transversal link susceptance.
Figure 2: With $v_0$ as the root, nodes $n_1$, $n_2$ and $n_3$ are downward of $m$ in all possible spanning trees of this network. Thus, the two components separated by $m$ (the biconnected components) can be optimized as independent problems.

If $n \in D_m'(T, v_0)$ and $m \in D_{uv}(T, v_0)$, then $n \in D_{uv}(T, v_0)$. If $n$ is downward of $m$ in all possible spanning trees, i.e., if $n \in D'_m(T, v_0)$, then $n$ is downward of $(u, v)$ iff $m$ is also downward of $(u, v)$ regardless of the spanning tree considered. Under this assumption, if $n$ is downward of $(u, v)$ then there is no change in the current $I_{uv}$ of any link $(u, v)$ not placed in the path between $m$ and $n$, since

$$I_{uv} = \sum_{k \in D_{uv}(T, v_0)} I_k^L = \sum_{k \in D_{uv}(T, v_0) \setminus \{n\}} \begin{cases} I_m^L + I_n^L & \text{if } k = m \\ I_k^L & \text{otherwise} \end{cases} \forall T \in \text{ST}(G).$$

This procedure can be repeated with $m$ for all nodes in the same condition as $n$. Also, from (3) one can see that $I_{uv}$ is independent of the tree path from the root node to link $(u, v)$. Thus, more generally, if a node $m$ is a cutvertex separating $G$ into two of more connected components, each component can be optimized as a separate problem. For the component including the root node, the total load of the other components is added to the load of $m$. For the other components, $m$ is considered their root node. If some of these components are trees, then there is no optimization to be done on those components. Figure 2 represents a network which can be partitioned into two components.

An equivalent statement can be made in terms of the biconnected components of $G$ – each of these components can be treated as a separate optimization problem, excluding the trivial components consisting of a dyad (i.e., a pair of nodes with a link between them). The graph of the 33-node test network (Figure 1) has two biconnected components: one is the dyad containing the link $(0,1)$, and the other one contains the remaining links with node 0 removed. The network to be optimized reduces to this second component, thus node 0 and link $(0,1)$ can be removed and node 1 becomes the new root node.

2.2 Spanning tree reduction through edge lifting

Let $G_C = (V_C, E_C)$ be a non-trivial biconnected component of $G$. $G_C$ can be reduced to a topological minor $G_0 = (V_0, E_0)$ where $V_0 \subset V_C$ through successive application of edge liftings – the two edges incident to a vertex with degree 2 are replaced by a single edge incident to the two neighbors of that vertex while the vertex is removed [15]. This step is repeated for all vertices of $G_C$ with degree 2 with the exception of the root vertex $v_0 \in V_C$ (if it also has degree 2) since $v_0$ must also be in $V_0$. Given that $G_C$ is a non-trivial biconnected component, its vertices have a degree of at least 2. With the possible exception of the root vertex, the vertices of $G_0$ have a degree of at least 3 – keeping the same degree as they have in $G_C$ – since $G_0$ contains no vertices with degree 2.

As a result of the edge lifting transformation, each edge $(u, v) \in E_0$ represents a path of $G_C$ with one or more edges. Let $P_{uv} \subset G_C$, with $V(P_{uv}) \cap V_0 = \{u, v\}$, be such a path and let $T_C$
be a spanning tree of $G_C$. If all edges of $P_{uv}$ are in $E(T_C)$ (i.e., if $P_{uv} \subset T_C$), then the path is said to be closed in $T_C$; otherwise the path is open in $T_C$. If a path is open, then exactly one of its edges is not in $E(T_C)$ and that edge is said to be the open edge of the path in $T_C$ – if two or more edges in the path would not be in $E(T_C)$, then $T_C$ would be either a disconnected graph or not spanning to all vertices of $G_C$, and thus $T_C$ could not be a spanning tree of $G_C$.

Let $T_0$ be a spanning tree of $G_0$. This tree corresponds to a $T_C$ such that any edge in $E(T_0)$ represents a closed path in $T_C$ and any edge in $E_0 \setminus E(T_0)$ represents an open path in $T_C$, i.e., $(u, v) \in E(T_0) \Leftrightarrow P_{uv}(G_C) \subset T_C$, $\forall (u, v) \in E_0$. Thus, a given $T_0$ corresponds to all spanning trees of $G_C$ with a given set of open (or closed) paths – the only difference between those trees is the set of open edges $Q = E_C \setminus E(T_C)$ belonging to those open paths.

With this method, the problem of finding the optimal spanning tree $T^*_C$ is equivalent to finding the optimal pair composed of the smaller spanning tree $T_0$ and the set $Q$ containing an open edge $q_{ab} \in E(P_{ab})$ for each edge $(a, b) \in E_0 \setminus E(T_0)$. The new problem is then given by

$$\left(T_0, Q\right)^* = \arg\min_{T_0 \in ST\left(G_0\right)} \sum_{(u,v) \in E(T_C(T_0,Q))} R_{uv} |I_{uv}(T_C(T_0,Q))|^2 \quad (5)$$

where $T_C$ is computed from the complete configuration $(T_0, Q)$ and $I_{uv}(T_C)$ is defined as in (3).

As most of the nodes of the main biconnected component of the 33-node test network (figure 1) have degree 2, this reduction yields a much smaller topological minor, as shown in figure 3. This reduced component has 463 distinct spanning trees, which compares with the 50571 spanning trees of the original network.

### 3 Problem Constraints

This section expresses the problem constraints in terms of binary variables (assuming the values 0 or 1) which will be part of the final QUBO model. These constraints fall into two main categories: topology constraints, which impose that the problem solution is a spanning tree, and auxiliary variables constraints, which assign values to auxiliary variables needed for the QUBO cost function (in this case, to express the network power losses). The first category encompasses vertex, edge, cycle and path constraints.

For the formulation of the problem, it is necessary to assign a direction to the edges of the spanning tree. This direction is defined as pointing downwards, i.e., away from the root vertex of the tree. The resulting directed spanning tree is thus a spanning arborescence composed of arcs (i.e., directed edges).
Let $G_0 = (V_0, E_0)$ be the undirected graph of a reduced non-trivial biconnected component of the original network graph (as described in 2.2) and let $v_0 \in V_0$ be the root of $G_0$. Let $G_D = (V_0, A)$ be the directed graph containing an arc for each valid direction of each edge of $G_0$. Given our definition for edges direction, the edges incident to the root have only one valid direction – away from the root. Thus, each of these edges corresponds to a single arc on $A$, and since the root has no incoming arc, it is also the source of $G_D$. All other edges admit the two directions depending on the spanning tree, thus assigning two arcs on $A$ with opposite directions for each edge. The set of valid arcs for the undirected edge $(u, v) \in E_0$ is then given by

$$A_{uv} = \begin{cases} 
\{(u, v)\} & \text{if } u = v_0 \\
\{(v, u)\} & \text{if } v = v_0 \\
\{(u, v), (v, u)\} & \text{otherwise}
\end{cases}, \quad \forall (u, v) \in E_0 \quad (6)$$

and

$$A = \bigcup_{(u, v) \in E_0} A_{uv}. \quad (7)$$

Let $T_D = (V_0, A_T)$ be a spanning arborescence of $G_D$, where $A_T \subseteq A$. For each arc $(u, v) \in A$, we define a binary variable $e_{uv}$ specifying if the arc is in $A_T$, thus following the definition for the same variable in the QUBO formulation for the Degree-Constrained Minimum Spanning Tree of [14]. In order to $T_D$ be a valid spanning arborescence, all of the following constraints must be met.

### 3.1 Vertex constraints

Each vertex in a spanning arborescence has exactly one incoming arc, with the exception of the root, as specified by the constraint

$$\sum_{u \in N_{G_0}(v)} e_{uv} = 1, \quad \forall v \in V_0 \setminus \{v_0\} \quad (8)$$

where $N_{G_0}(v)$ is the set of all neighbours of $v$ in $G_0$. The vertex constraints also follow the same definition as in [14].

### 3.2 Edge constraints

It is not possible to have in an arborescence both arcs of an undirected edge since this would close a 2-cycle. This condition is prevented with the constraint

$$e_{uv} e_{vu} = 0 \quad \forall (u, v) \in E_0, \ u, v \neq v_0. \quad (9)$$

### 3.3 Cycle constraints

The previous two types of constraints impose that the number of arcs is equal to the number of vertices minus one. While this condition is necessary to form a spanning arborescence, it is not sufficient since the selected arcs may, with the previous constraints alone, still close directed cycles in $G_D$ and, equivalently, separate $T_D$ into a disconnected subgraph. Thus, additional constraints are needed to prevent the closing of any possible cycle.

Since $G_0$ is a non-trivial biconnected component, each of its vertices and edges is in at least one cycle. Assuming that $G_0$ is planar, as is typically the case of electrical distribution grids, the
set of its facial cycles, excluding the outer face cycle, define a cycle basis in $G_0$. Given that the root has no incoming arc in $G_D$, no directed cycle containing the root can be closed in $G_D$ since such a cycle would require at least one incoming arc (and one outcoming arc) incident to that vertex. Thus, the facial cycles containing the root are excluded from the cycle basis considered for the cycle constraints. Figure 4 shows the four cycles (A, B, C and D) of the cycle basis of the reduced main biconnected component of the 33-node test network.

3.3.1 Single cycle constraints

There is a directed cycle in $G_D$ for each of the two possible directions of a cycle in $G_0$. A constraint is needed to prevent the closure of each directed cycle, as specified by

$$\prod_{i=1}^{n-1} e_{k_ik_{i+1}} = 0 \quad (10a)$$

$$\prod_{i=1}^{n-1} e_{k_{i+1}k_i} = 0 \quad (10b)$$

where $k_{n+1} \equiv k_1$ for a $n$-length cycle in $G_0$ given by the sequence of vertices $k_1, k_2, \ldots, k_n, k_1 \in V_0$.

3.3.2 Combined cycle constraints

Let $C_1$ and $C_2$ be two cycles of the cycle basis sharing a contiguous path composed of one or more common edges and let $C_{12}$ be the combination of those two cycles. $C_{12}$ is also a cycle and it is induced by the symmetric difference of the edge sets of $C_1$ and $C_2$, i.e., the union of these sets excluding the common edges.

Although the single cycle constraints applied for $C_1$ and $C_2$ prevent the closing of those cycles, this is not the case with $C_{12}$ since the arcs of the common edges of $C_1$ and $C_2$ would not be in $A_T$ and thus forcing the product of the above constraints to be zero for both $C_1$ and $C_2$. As a consequence, $C_{12}$ would have to be considered as an additional cycle on those constraints in order to prevent its closure.

With multiple adjacent cycles, constraints must be added to each possible cycle combination. Figure 5 shows the adjacency between the cycles of the cycle basis shown in figure 4. There are eight cycle combinations for this basis: AB, ABC, ABD, ABCD, BC, BD, CD and BCD.
3.3.3 A new method for adjacent cycles

As previously shown, the number of adjacent cycle combinations in our test network is twice the number of cycles in the cycle basis. For larger bases, the number of combinations grows more than linearly with the basis size, resulting in an even larger number of additional cycle constraints being added to the model.

We are proposing a new method for adjacent cycle constraints without the need to consider the cycles formed by combinations of adjacent cycles. This method is based on the idea of adding an auxiliary direction variable \( d_{uv} \) to each edge \((u, v)\) common to two adjacent cycles \(C_1\) and \(C_2\). This variable assigns a direction to the edge even if none of its two arcs is in \(A_T\) (and thus defining a direction). Since \((u, v)\) is an undirected edge, \(d_{uv}\) is defined for the convention \(u < v\).

Its constraints are given by

\[
\begin{align*}
e_{uv} & \implies d_{uv} & \quad (11a) \\
e_{vu} & \implies \neg d_{uv} & \quad (11b)
\end{align*}
\]

where \(\neg d_{uv} \equiv 1 - d_{uv}\) is the logical negation of \(d_{uv}\). For this edge, these constraints replace the edge constraint (9) since they already prevent \(e_{uv}\) and \(e_{vu}\) from being both true.

From these constraints it is clear that if no arc of the edge \((u, v)\) is in \(A_T\) then the value of \(d_{uv}\) is not imposed by these constraints. This is the case when \(C_1\) and \(C_2\) are not closed but \(C_{12}\) is. To prevent the closure of \(C_{12}\) while still preventing \(C_1\) and \(C_2\) from closing, the single cycle constraints (10) for these two cycles must be changed by replacing \(e_{uv}\) with \(d_{uv}\) or \(\neg d_{vu}\), as given by

\[
\begin{align*}
\prod_{i=1 \ldots n} e'_{k_i k_{i+1}} &= 0 & \quad (12a) \\
\prod_{i=1 \ldots n} e'_{k_{i+1} k_i} &= 0 & \quad (12b)
\end{align*}
\]

where \(e'_{uv} \equiv \begin{cases} d_{uv} & \text{if } u < v \land (u, v) \in E(C_1) \cap E(C_2) \\ \neg d_{vu} & \text{if } v < u \land (u, v) \in E(C_1) \cap E(C_2) \\ e_{uv} & \text{otherwise} \end{cases}\).

Given the logical implications of (11), it is clear that (12) still prevents the closure of \(C_1\) and \(C_2\). The closure of \(C_{12}\) is also prevented since one of the constraints for either \(C_1\) or \(C_2\) would be violated, depending on the value of \(d_{uv}\) which, in this situation, is not imposed by (11). In other words, (12) together with (11) are stricter than (10) alone because every common edge must have a direction regardless of having any of its arcs in \(A_T\) or not, and that direction will violate one of the cycle constraints if \(C_{12}\) is closed, as illustrated in figure 6.
3.3.4 Generalization to cycles with interior vertices

While an edge cannot be in more than two facial cycles, a vertex can be shared by more than two of such cycles. This vertex may lie in the outer face cycle (which is not part of the cycle basis), or otherwise be an interior vertex. As mentioned in 2.2, all graph vertices after edge lifting have a degree of at least 3, excluding the root vertex. Then, any interior vertex is necessarily in three or more cycles. In our test network cycle basis, shown in figure 4, vertex 8 is the only interior vertex and thus it is the only vertex common to at least three cycles (namely, cycles B, C and D).

The closure of a combination of cycles containing all cycles sharing an interior vertex – as is the case of combinations BCD and ABCD in our test network – is not prevented by the constraints defined in 3.3.3 alone. To handle this case, a virtual cycle must be added such that it encloses the interior vertex. The cycle vertices are the neighbors of the interior vertex and the cycle edges may or may not be in $E_0$. If any of these neighbors is also an interior vertex, a new virtual cycle is added to enclose this vertex, and the procedure is repeated until there are no further interior vertices to enclose. To illustrate this method with the test network, a virtual cycle is added to enclose the interior vertex 8, as shown in figure 7.

The new virtual cycle $V$ is defined by the neighbors of vertex 8 – vertices 7, 11 and 14 – and the following relations hold:

$$V = C \oplus B_2 \oplus D_2$$

$$B = B_1 \oplus B_2$$

$$D = D_1 \oplus D_2$$

(13)
where ⊕ represents the combination of two cycles. From the three edges of cycle V, only the edge (11,14) is in \( E_0 \). Since this edge is common to two cycles (V and C), a \( d_{11,14} \) variable is allocated as previously described. The other two edges, represented by the dashed lines in figure 7, are shared by three cycles each – edge (7,14) is common to cycles V, B and \( E_1 \) allocated as previously described. The other two edges, represented by the dashed lines in figure 7, are shared by three cycles each – edge (7,14) is common to cycles V, B1 and B2, while edge (7,11) is common to V, D1 and D2. Variables \( d_{7,14} \) and \( d_{7,11} \) are also allocated for these two edges, but since these edges are not in \( E_0 \), no \( e \) variables are allocated to them and thus the constraints defined in (11) do not apply to these edges.

All cycles in figure 7 except V are facial cycles, and thus they compose a valid cycle basis. Applying the constraints defined in (12) to these facial cycles alone would yield the same results as with the original cycle basis, where the closure of \((A)BCD\) is not prevented. On the other side, cycle V may be viewed as a shorter representation of \((A)BCD\) since those constraints force the closure of the former if and only if the latter is closed. Then, applying these constraints to cycle V as well will finally prevent the closure of \((A)BCD\).

### 3.4 Path constraints

The edge lifting reduction described in section 2.2 assigns a path \( P_{uv} \subset G_C \) for each edge \((u,v)\) of the topological minor \( G_0 \subset G_C \), such that a network configuration in \( G_C \) can be represented by the pair \((T_0,Q)\), where \( T_0 \) is a spanning tree of \( G_0 \) and \( Q \) is the set of open edges of \( G_C \) (i.e., the edges not in the spanning tree \( T_C \) equivalent to \((T_0,Q)\)). A network configuration can be equally represented by the spanning arborescence \( T_D \) instead of \( T_0 \) in the pair \((T_D,Q)\).

In order to indirectly specify the set \( Q \), a binary variable \( p_{x} \) is added for each inner vertex \( x \) of a path \( P_{uv} \) (thus excluding the end vertices \( u \) and \( v \)), for all edges of \( G_0 \). Since \( P_{uv} \) is undirected, the convention \( u < v \) is assumed in the following discussion. Let the path \( P_{uv} \) with \( n \) inner vertices be represented by the vertex sequence \( u, k_1, k_2, \ldots, k_n, v \). The \( p_{k_i} \) variable specifies which of the end vertices – \( u \) (if 1) or \( v \) (if 0) – is upward of the inner vertex \( k_i \), with \( i = 1 \ldots n \).

If the path is open, let \( k_i \) and \( k_{i+1} \) be the end vertices of the open edge within the path. Then, the inner vertices \( k_1, \ldots, k_i \) have \( u \) as the upward end vertex, and the inner vertices \( k_{i+1}, \ldots, k_n \) have \( v \) as the upward end vertex. Thus, the open edge is the only one in the path where \((p_{k_i}, p_{k_{i+1}}) = (1,0), \exists i = 1, \ldots, n-1 \). Also, the condition \((p_{k_i}, p_{k_{i+1}}) = (0,1), \exists i = 1, \ldots, n-1 \) is never allowed in any path, as specified for all paths \( P_{uv} \) with \( n_{uv} > 1 \) inner nodes by the constraint

\[
P_{n_{uv}} \Rightarrow p_{k_i^{uv}}, \quad \forall (u,v) \in E_0, u < v, i = 1, \ldots, n_{uv} - 1
\]  

(14)

where \( k_i^{uv} \) is the \( i \)-th inner node of path \( P_{uv} \) counted from \( u \) to \( v \).

If a path \( P_{uv} \) is closed, either \( e_{uv} \) or \( e_{vu} \) specifies the path direction and all the path inner vertices have the same upward end vertex, either \( u \) or \( v \) respectively, as specified in (14) together with the constraints

\[
e_{uv} \Rightarrow p_{k_i^{uv}}
\]

(15a)

\[
e_{vu} \Rightarrow \neg p_{k_i^{uv}}
\]

(15b)

defined for any path \( P_{uv} \) with \( n_{uv} > 0 \) inner nodes.

### 3.5 Auxiliary load-arc variables

For the sake of clarity in the discussion, this section first explains the method for the auxiliary load-arc variables assuming that \( G_0 \equiv G_C \), i.e., that no edge lifting was performed to transform \( G_C \) into \( G_0 \). Later, the edge lifting transformation is then considered.
3.5.1 Formulation without edge lifting

As formulated in (3), the set $D_{uv}(T, v_0)$ (of all vertices downward of a link $(u, v) \in E(T)$ across the spanning tree $T$) is needed in order to compute the electrical current flowing through the link. Given that the problem constraints are formulated in terms of a spanning arborescence $T_D = (V_0, A_T)$, we consider instead the set $D_{uv}(T_D, v_0)$ of all vertices downward of an arc $(u, v) \in A_T$ across $T_D$. An auxiliary binary variable $z_{uvn}$ is defined to specify whether a vertex $n$ is in $D_{uv}(T_D, v_0)$, i.e., whether the load of $n$ contributes to the current on the arc $(u, v)$ (thus giving the name load-arc variables). These auxiliary variables are needed to express the network power losses in the QUBO model, as will be described in the next section.

Since $A_T$ and thus $T_D$ are defined by the $e_{uv}$ variables previously introduced, each $z_{uvn}$ variable is ultimately a function of $e$ variables. Assigning a $z_{uvn}$ variable from $e$ variables alone would lead to extremely complex boolean expressions since all possible paths in $G_D$ between vertices $v$ and $n$ would need to be explicitly accounted for. This would defeat the whole purpose of the QUBO optimization since the complete search space would need to be traversed just to build the QUBO model.

We are proposing a novel method to assign the $z$ variables with minimal overhead. The central concept of this method is having these variables depending not only on $e$ variables but on other neighboring $z$ variables as well. As an illustration of this method, vertex $n$ in figure 8 contributes to the current in arc $(u, v)$ (as specified by $e_{uv}$) if and only if the arc is in $T_D$ (as stated by $e_{uv}$) and if the vertex also contributes to the current in arcs $(v, m_2)$ or $(v, m_3)$. Since no cycles are allowed in the arborescence, $n$ contributes to at most one of these two arcs. The complete definition is then $z_{uvn} = e_{uv}(z_{vm2n} + z_{vm3n})$. Similarly, for the arc in the opposite direction, $z_{vun} = e_{vu}(z_{um1n} + z_{um2n})$.

The $z$ variables are then more generally defined as

$$z_{uvn} = e_{uv} \sum_{m \in N_{G_D}(v) \setminus \{u, v_0\}} z_{vmn} \quad \forall (u, v) \in A, n \in V_0 \setminus \{u, v, v_0\}. \quad (16)$$

One may notice that this formulation applied to the given example includes $z_{vtn}$ and $z_{usn}$ in the definition of $z_{uvn}$ and of $z_{vun}$, respectively, but it is easy to realize that both $z_{vtn}$ and $z_{usn}$ are always zero. This formulation does not define $z_{uvn}$ if $n = v$, but in the expression $z_{vnm}$, $m$ may be equal to $n$. Thus, to define a $z_{xyy}$ variable, one may realize that it is always equal to $e_{xy}$ and in fact they are one and the same variable in the QUBO model, as stated by

$$z_{uvv} \equiv e_{uv}, \quad \forall (u, v) \in A. \quad (17)$$
3.5.2 Formulation with edge lifting

As defined in (16), a $z_{uvn}$ variable is allocated for each combination of an arc in $A$ with all vertices of $V_0$ excluding the arc end-vertices and $v_0$, thus the number of such variables is $|A|(|V_0| - 3) + d_{G_0}(v_0)$, where $d_{G_0}(v_0)$ is the degree of $v_0$ in $G_0$. The application of (16) and (17) to the main biconnected component of our test network without edge lifting would result in 2032 $z$ variables. Given the size limits of current quantum annealers, it is crucial to reduce the number of these variables. This is the main motivation to use edge lifting to reduce $G_C$ into the topological minor $G_0$.

Since an arc $(u, v) \in A$ represents a path $P_{uv} \subseteq G_C$ where each link in the path has a distinct current value, in general there is no single current value on the arc (unless the path has one link only). Thus, with edge lifting, a $z_{uvn}$ variable shall be interpreted as whether a vertex $n \in V_C$ contributes to the current of the links in path $P_{uv}$ along the direction defined by arc $(u, v)$.

Given that $V_0 \subset V_C$, (16) and (17) applied to $G_D$ would not define a $z_{uvn}$ variable (and thus a $z_{vmn}$ variable) when vertex $n \in V_C$ and $n \notin V_0$, although its load current would also contribute to the current of some network links. Such a vertex is necessarily an inner vertex of some path $P_{uv}$ where $(u, v) \in E_0$. In this case, just like a $z_{uvw}$ variable, $z_{uvn}$ does not exist as an explicit QUBO variable. Instead, it is defined as

$$z_{uvn} = \begin{cases} p_n & \text{if } u < v \\ \neg p_n & \text{if } u > v \end{cases} \forall (u, v) \in \bigcup_{(a, b) \in E_0} \{(a, b), (b, a)\}, \ n \in V(P_{uv}) \setminus \{u, v\}. \ (18)$$

This definition applies to both directions of every edge in $E_0$, instead of being applied to all arcs in $A$, since $z_{uvn}$ variables are used in the power losses function and this function applies to both directions of every path $P_{uv}$ including the ones incident to the root vertex, as will be seen in 4.2.

Finally, (16) must be generalized to define $z_{uvn}$ when vertex $n \in V_C$, $n \notin V_0$ is an inner vertex of a path other than $P_{uv}$, as given by

$$z_{uvn} = c_{uv} \sum_{m \in NC_0(v) \setminus \{v_0\}} z_{vmn} \quad \forall (u, v) \in A, \ n \in V_C \setminus (V(P_{uv}) \cup \{v_0\}). \ (19)$$

With (17), (18) and (19), $z_{uvn}$ is now defined for any arc $(u, v) \in A$ and for any vertex $n \in V_C \setminus \{u, v_0\}$. The hypothetical $z_{uvv}$ and $z_{uvv0}$ variables would always be zero and thus they do not need to be defined. With edge lifting, the number of $z$ variables for the main biconnected component of our test network reduces to 654.

A $z_{uvn}$ variable is always zero when, due to the network topology, there is no spanning arborescence such that vertex $n$ contributes to the current in arc $(u, v)$. Such $z$ variables can be excluded from the model, resulting in a further reduction of the number of $z$ variables to 577.

4 QUBO Formulation

A QUBO model formulates a pseudo-Boolean function $f: \mathbb{B}^n \to \mathbb{R}$ [16] as a quadratic polynomial over $n$ binary variables $x_i \in \mathbb{B}$, where $\mathbb{B} = \{0, 1\}$:

$$f(x) = \sum_{i=1}^{n} a_i x_i + \sum_{i<j} b_{ij} x_i x_j + c \quad (20)$$

where $a_i$ and $b_{ij}$ are the real-valued linear and quadratic coefficients, respectively, and $c$ is a constant term. Solving a QUBO problem consists on finding the binary string $x^*$ which minimizes
While the constant term $c$ has no influence on $x^*$, its inclusion on $f$ makes this function more general in order to provide meaningful cost values for the optimization problem at stake.

For our optimization problem, the solution $x^*$ contains the variables described in the previous section. Thus, each of these variables is assigned to a given variable index $i$ in the QUBO model.

4.1 Problem constraints

Since, by definition, the QUBO model is unconstrained, the problem constraints described in the previous section must be added to the model as penalty expressions. To this end, these constraints are first written as a Constraint Satisfaction Problem (CSP) using the `dwavebinarycsp` Python package from the D-Wave Ocean SDK [17]. The CSP is then converted to the QUBO model using the `stitch` function from this package with the default minimum penalty value of 2.0. Thus, the violation of any constraint would increase the value of the QUBO model by at least 2.0, while all feasible solutions keep the QUBO value unchanged. The QUBO model is represented by a `BinaryQuadraticModel` class instance on which the power losses terms are added.

4.2 Power losses function

Network power losses minimization is the goal of this constrained optimization problem. As defined in 2.2, let $G_0 = (V_0, E_0)$ be a topological minor of a non-trivial biconnected component $G_C = (V_C, E_C)$ of the original network $G$. A power losses function can be defined for each edge $(u, v) \in E_0$ as the sum of the losses in all links $(a, b)$ of the path $P_{uv} \subset G_C$. This function depends on the network configuration, which can be represented by the spanning tree $T_C$ of $G_C$, as given by

$$L^0_{uv}(T_C) = \sum_{(a, b) \in E(P_{uv})} L_{ab}(T_C) \quad \forall (u, v) \in E_0$$

where the losses function $L_{ab}(T_C)$ is defined as in (2). Given that each link $(a, b) \in E_C$ is in exactly one path $P_{uv}$, the total network losses $\sum_{(a, b) \in E_C} L_{ab}$ are same as $\sum_{(u, v) \in E_0} L^0_{uv}$ and thus the original optimization problem defined in (1) for the component $G_C$ can be reformulated in terms of $L^0_{uv}$.

With the auxiliary load-arc variables introduced in the previous section, the power losses function can be expressed as a sum of linear and quadratic terms of these variables and thus these terms can be directly added to the QUBO model. As formulated in (2), the power losses in a given link are quadratic with the current flowing on the link and, as stated in (3), this current is the sum of the currents from all loads downward of the link. Thus, the expansion of the square of the sum yields quadratic terms for all pairwise combinations between the sum terms.

The losses $L^0_{uv}$ can defined as the sum of two directed losses

$$L^0_{uv} = L^D_{uv} + L^D_{vu} \quad \forall (u, v) \in E_0$$

where $L^D_{uv}$ represents the total losses on the edges of $P_{uv}$ downward of vertex $u$ and $L^D_{vu}$ corresponds to the losses on the edges downward of $v$. If the path is closed then all the links in the path are downward of either $u$ or $v$ and then one of the directed losses is zero. The directed losses in an arc $(u, v)$ of each of the two directions of an edge in $E_0$ can be expressed in terms of
QUBO variables, as given by

\[ L^D_{uv} = c_{uv} \left( R_{uv} \left| I^L_n \right|^2 + L'_{uv} (I^L_n) \right) \]

\[ + \sum_{n \in V(P_{uv}) \setminus \{u,v\}} z_{u vn} \left( R_{un} \left| I^L_n \right|^2 + \sum_{k \in V(P_{uv}) \setminus \{u,n\}} 2R_{uk} \Re \left( \frac{I^L_n}{I^L_k} \right) \right) \]

\[ + \sum_{n \in V \setminus (V(P_{uv}) \cup \{v\})} z_{u vn} \left( R_{uv} \left| I^L_n \right|^2 + 2 \Re \left( \frac{I^L_n I^L_u}{I^L_y} \right) \right) + L'_{uv} (I^L_n) \]

\[ + \sum_{n,k \in V \setminus (V(P_{uv}) \cup \{v\}) \setminus \{u\}} z_{u vn} z_{u vk} 2R_{uv} \Re \left( \frac{I^L_n}{I^L_k} \right) \]

where

\[ L'_{uv} (I) \equiv \sum_{k \in V(P_{uv}) \setminus \{u,v\}} 2R_{uk} \Re \left( \frac{I^L_n}{I^L_k} \right) \]

\[ R_{ux} \equiv \sum_{(a,b) \in E(P_{uv})} R_{ab} \]

with \( \Re(c) \) and \( \Re(c) \) being the real part and the complex conjugate, respectively, of a complex number \( c \), and with \( P_{uv} \subseteq P_{uv} \) being the path along \( P_{uv} \) between vertices \( u \) and \( x \) where \( x \in V(P_{uv}) \setminus \{u\} \).

Given the set \( V(P_{uv}) \setminus \{u,v\} \) for the vertex \( n \) in the first sum of the second line of (24), the \( z_{u vn} \) variables in this line are defined in (18) as being equivalent to either \( p_n \) or \( \neg p_n \), while the remaining \( z \) variables in (24) are defined in (19) as explicit variables. Note that \( L^D_{uv} \) is defined for any arc \((u,v) \in A \) \( (a,b) \in E \) \( (a,b), (b,a) \) \( \supseteq A \). If \( v = v_0 \) in such an arc, then \((u,v) \notin A \) and all \( e \) and explicit \( z \) variables in (24) are not defined and thus they are considered to be zero. As a consequence, only the terms in the second line of (24) are nonzero in this situation.

Before being added to the QUBO model, the losses terms are scaled by a suitable constant factor. Applying a constant scaling to the network losses does not change the optimal solution of the original problem defined in (1). The purpose of this scaling is to keep the optimal losses value below the minimum constraints penalty value of 2.0 (mentioned in Section 4.1), while maximizing the magnitude of the losses function to gain resolution on the quantum annealer since it has a limited dynamic range for the physical realization of the model coefficients [18]. If the scaled optimal losses value were greater than the minimum constraints penalty value, the solution with minimum QUBO value could be infeasible since a constraint could be violated in order to unfeasibly decrease the network losses.

5 Results

In this section the results of the application of our formulation are presented and discussed. These results include the QUBO model metrics for the 33-node test network and the validation of this model through the analysis of its optimal solution as found by classical solvers.

5.1 QUBO model metrics

The QUBO model for the 33-node test network has a total of 1074 variables, as detailed in table 1. Variables \( e, d, z \) and \( p \) were already described in section 3. The \( y \) variables were manually
| Variable class | Number of variables |
|----------------|---------------------|
| e              | 24                  |
| d              | 4                   |
| p              | 23                  |
| z              | 577                 |
| y              | 434                 |
| auxiliary      | 12                  |
| **Total**      | **1074**            |

Table 1: QUBO model variable allocation per variable class.

| Constraints type / Function | Interactions |
|-----------------------------|--------------|
| Vertex (8)                  | 24           |
| Edge (9) or (11)            | 13           |
| Cycle (12)                  | 72           |
| Path (14)                   | 14           |
| Edge-path (15)              | 17           |
| z and y variables (19)      | 3029         |
| Network losses (24)         | 6997         |
| **Total**                   | **10166**    |

Table 2: QUBO model interactions count per constraints type or per function.

introduced to improve the conversion to QUBO of some \( z \) variables constraints defined in (19) by reducing the number of interactions between variables. Finally, the auxiliary variables are automatically added in the conversion to QUBO of the cycle constraints defined in (12) which involve the product of three or more variables.

The model has a total of 10166 interactions between variables (i.e., product terms between two variables), as detailed in table 2. As formulated in (24), the \( z \) variables are needed for the network losses terms. Together with their intermediary \( y \) variables, they represent 94.1% of the model variables, while their interactions within constraints (19) and network losses (24) account for 98.6% of the model interactions. The dominance of these variables in the model size justifies the effort in reducing the number of \( z \) variables. As described in 3.5.2, this number was reduced from 2032 to 654 thanks to edge lifting and further reduced to 577 through elimination of null \( z \) variables.

Figure 9 shows the distribution of the number of interactions per variable. This metric is important for the embedding of the QUBO model in a quantum annealer [19]. This distribution spans from 3 to 71 interactions per variable, with an average of 18.9 and a sharp peak of 436 variables with 4 interactions each. Considering the variable classes, this distribution is partitioned into relatively well defined regions, as shown in table 3.

5.1.1 Comparison with other formulations

Our model cannot be completely compared with other QUBO formulations found in the literature since no other known formulation includes an optimization objective with a quadratic cost over the network flows. Still, the part of the model relative to the constraints ensuring a valid spanning tree solution can be compared with the equivalent function of other formulations. Tables 4 and 5 compare the number of variables and the number of interactions, respectively, needed for such constraints between spanning tree problem formulations found in the literature and our
Figure 9: Distribution of the number of interactions per model variable

| Variable class | Interactions per variable (number of variables) |
|----------------|-----------------------------------------------|
| auxiliary      | 3(10), 4(2)                                  |
| y              | 4(434)                                        |
| p              | 5(1), 6(1), 10(21)                            |
| d              | 8(1), 10(1), 13(2)                            |
| z              | 14...34(577)                                  |
| e              | 31(1), 35...71(23)                            |

Table 3: Distribution of the number of interactions per variable for each variable class.
on the linear chains of this graph. The advantage with $G$ interactions for both graphs with a linear scaling while the other formulation shows a cubic scaling. Electrical networks. Both with $G$ and $G_C$, the metrics with effect of edge lifting (i.e., considering only vertex, edge and cycle constraints), while the metrics with $G$ have a quadratic scaling (assuming $O(|V|)$ equivalent to $O(|V|^2)$). Given than the average vertex degree is not expected to grow with larger models have the same definition for vertex constraints, the term $2|E| - |N_G(v_0)|$ is equal to $|e|$. The number of auxiliary variables in our model grows linearly with the total number of variables in cycle constraints (i.e., total number of arcs in basis cycles) in excess of three variables per constraint; thus a linear scaling with $|E|$ is assumed for $|auxiliary|$ as for $|e|$, $|d|$ and $|p|$.

| Model | Number of variables | With $G_0$ | With $G_C$ | Scaling |
|-------|---------------------|------------|------------|---------|
| [13] $^4$ | $|V|([|V| + 3]/2) + |E|(|V| + 1)$ | 184 | 1732 | $O(|V||E|)$ |
| [14] | $2|E| - |N_G(v_0)| + \left\lfloor \frac{|V|-1}{2} \right\rfloor$ | 52 | 535 | $O(|V|^2)$ |
| Ours | $|e| + |d| + |auxiliary|$ | 40 | - | $O(|E|)$ |
| Ours | $|e| + |d| + |p| + |auxiliary|$ | - | 63 | $O(|E|)$ |

Table 4: Number of variables needed for spanning tree constraints in $G_0$ and $G_C$. To enable a fair comparison, the terms of the degree constraints were removed from the first two formulations since our model does not include such constraints. Given that [14] and our model have the same definition for the $e$ variables, the term $2|E| - |N_G(v_0)|$ is equal to $|e|$. The number of auxiliary variables in our model grows linearly with the total number of variables in cycle constraints (i.e., total number of arcs in basis cycles) in excess of three variables per constraint, thus a linear scaling with $|E|$ is assumed for $|auxiliary|$ as for $|e|$, $|d|$ and $|p|$.

| Model | Number of interactions | With $G_0$ | With $G_C$ | Scaling |
|-------|------------------------|------------|------------|---------|
| [14] | $\sum_{v \in V \setminus \{v_0\}} (\left\lfloor \frac{|N_G^{(v)}|}{2} \right\rfloor) + 2(|E| - |N_G(v_0)|) + 3\left\lfloor \frac{|V|-1}{3} \right\rfloor$ | 214 | 13600 | $O(|V|^3)$ |
| Ours | $\text{NI(vertex con.)} + \text{NI(edge con.)} + \text{NI(cycle con.)}$ | 109 | - | $O(|E|)$ |
| Ours | same as above $+ \text{NI(path c.)} + \text{NI(edge-path c.)}$ | - | 140 | $O(|E|)$ |

Table 5: Number of interactions needed for spanning tree constraints in $G_0$ and $G_C$ for the formulation of [14] and for our model. No number of interactions was provided in [13]. As in table 4, the terms of the degree constraints were removed from the first formulation. $\text{NI}$ represents the number of interactions for the given constraints class in our model (table 2). Since both models have the same definition for vertex constraints, the term $\sum_{v \in V \setminus \{v_0\}} (\left\lfloor \frac{|N_G^{(v)}|}{2} \right\rfloor)$ is equal to $\text{NI(vertex con.)}$. Given than the average vertex degree is not expected to grow with larger networks, $O(|E|)$ and $O(|V|)$ are considered equivalent and a linear scaling with $|E|$ is assumed for $\text{NI(vertex con.)}$ as for $\text{NI(edge con.), NI(cycle con.), NI(path con.) and NI(edge-path con.)}$.

model. This comparison considers the non-trivial biconnected component $G_C$ of the 33-node test network and its reduced form $G_0$ obtained from the edge lifting procedure (figure 3), where $G_C = (V_C, E_C)$ and $G_0 = (V_0, E_0)$ with $|V_C| = 32$, $|E_C| = 36$, $|V_0| = 9$ and $|E_0| = 13$. The metrics with $G_0$ enable the comparison between formulations without taking into account the effect of edge lifting (i.e., considering only vertex, edge and cycle constraints), while the metrics with $G_C$ include this effect leveraging on the existence of linear chains which are typically found in electrical networks. Both with $G_0$ and with $G_C$, our model allocates less variables than the other formulations and it shows a linear scaling with the network size, while the other formulations have a quadratic scaling (assuming $O(|E|)$ equivalent to $O(|V|)$). Our model also allocates less interactions for both graphs with a linear scaling while the other formulation shows a cubic scaling. The advantage with $G_C$ is particularly striking given the additional effect of edge lifting on the linear chains of this graph.

$^3$The original expression in [13] has a minor mistake. The correct expression for the formulation of [13] shown here is actually given in [14] while citing that formulation.
5.1.2 Scaling of the complete model

The complete QUBO model includes the network losses terms which, as described in 5.1, dominate the model in terms of both the number of variables and the number of interactions. The $z$ variables – and their intermediary $y$ variables – are needed to compute the network losses as a QUBO value. As mentioned in 3.5.2, a variable $z_{uvn}$ is defined for every pair of an arc $(u, v) \in A$ with a vertex $n \in V_C \setminus \{u, v_0\}$. As defined in 3, $A$ is the set of arcs in $G_D$ (the directed graph associated with $G_0$). Given that $|A| \in \mathcal{O}(|E_0|)$, the number of $z$ variables is $\mathcal{O}(|V_C||E_0|)$. The number of $y$ variables has the same order.

The product terms of the network losses, as defined in the last line of (24), account for the great majority of the model interactions, as seen in table 2. For every arc $(u, v)$, each pair of distinct vertices $n, k \in V_C \setminus (V(P_{uv}) \cup \{v_0\})$ originates a product $z_{uvn}z_{uvk}$ in the model. Given that the number of products per arc is $\mathcal{O}(|V_C|^2)$, the total number of products is $\mathcal{O}(|V_C|^2|E_0|)$.

5.2 QUBO model validation

In order to validate our formulation, the optimal solution of the QUBO model for the 33-node test network was found by classical solvers and this solution was compared with a reference optimal solution found by exhaustive search over the complete spanning trees space of this network. This search computed the network losses as a direct application of (1) for each spanning tree of the unmodified network model. Thus, this solution does not depend on any network simplification or any QUBO construct as defined in the previous sections. The optimal solution found with this method is represented by the set of open links $\{(6,7),(8,9),(13,14),(31,32),(24,28)\}$. The remaining network links, and only these ones, belong to the spanning tree corresponding to this solution. This optimal configuration has a total power loss of 116.379 kW.

As mentioned in Section 4.2, the network losses terms in the QUBO model are scaled by a constant factor such that the optimal losses value sits below the constraints penalty threshold of 2.0. With these losses expressed in kW, this factor was chosen to be 0.01. Thus, the scaled QUBO value of the optimal solution is 1.16379.

The optimal solution for the QUBO model was found by several MIQP solvers from NEOS Server [20–22]: FICO Xpress [23] (in 48 seconds), CPLEX [24] (in about 5 minutes) and SCIP [25] (in about 23 minutes). All these solvers returned the expected solution, with the same set of open links and the same QUBO value.

Although this result is not a definitive guarantee of the model correctness, we strongly believe this is the case since it would be extremely unlikely that a mistake on the network losses terms would result in the same computed value for the losses. Additionally, if the model constraints were insufficient to impose a correct network topology, the optimal solution of the QUBO model would result in a disconnected network in order to unfeasibly decrease the losses value.

6 Conclusion

In this paper, a new QUBO model for the minimum loss reconfiguration problem was proposed. A comparison with other QUBO formulations of spanning tree reconfiguration problems showed that our formulation is more efficient in terms of the number of variables and interactions. This result holds not only for the 33-node test network used as an illustrative example, but also for larger networks given the formulation scaling. This efficiency is given by our general topology constraints – vertex, edge and cycle constraints – but it is also a result of the edge lifting transformation, which takes advantage of the linear chains typically found in electrical networks.
Our topology constraints can be used in other QUBO formulations of spanning tree problems in order to obtain a smaller model, particularly for sparse graphs.

The paper has also shown that the remaining part of our model – the network losses function – dominates the QUBO model in terms of the number of variables – with a quadratic scaling with the network size – and in terms of the number of interactions – with a cubic scaling. Additionally, it was described how edge lifting reduces the number of network losses variables to less than one third. The number of interactions for the losses function is reduced by the same ratio, given the linear scaling of this number with the number of edges of the reduced graph.

The optimal solution for the QUBO model of the example network was obtained from several solvers. This solution was found to be the same as the one obtained from exhaustive search with a non-QUBO formulation of the same problem.

With the results obtained for the 33-node network, we expect that our new QUBO formulation will enable the use of quantum annealing or quantum-classical solvers for handling reconfiguration problems on real-world electrical networks with advantage over classical solvers in terms of solution quality and time-to-solution. This advantage will hopefully become more apparent as quantum annealers and hybrid solvers will continue to improve their performance.

Acknowledgements

This work was partially supported by Portuguese national funds through FCT – Fundação para a Ciência e a Tecnologia with references UIDB/50021/2020 and UIDB/50008/2020. FFCS thanks the support from Fundação para a Ciência e a Tecnologia and from the European Social Fund through the scholarship SFRH/BD/143402/2019.

References

[1] Filipe F. C. Silva, Pedro M. S. Carvalho, and Luís A. F. M. Ferreira. “Improving PV Resilience by Dynamic Reconfiguration in Distribution Grids: Problem Complexity and Computation Requirements”. In: Energies 14.4 (2021). ISSN: 1996-1073. DOI: 10.3390/en14040830.

[2] Mesut E. Baran and Felix F. Wu. “Network reconfiguration in distribution systems for loss reduction and load balancing”. In: IEEE Transactions on Power Delivery 4.2 (1989), pp. 1401–1407. DOI: 10.1109/61.25627.

[3] Sivkumar Mishra, Debapriya Das, and Subrata Paul. “A comprehensive review on power distribution network reconfiguration”. In: Energy Systems 8.2 (2017), pp. 227–284. ISSN: 1868-3975. DOI: 10.1007/s12667-016-0195-7.

[4] Pedro M. S. Carvalho, Luís A. F. M. Ferreira, and Alexandre M. F. Dias. “Distribution grids of the future: Planning for flexibility to operate under growing uncertainty”. In: Foundations and Trends® in Electric Energy Systems 2.4 (2018), pp. 324–415. ISSN: 2332-6557. DOI: 10.1561/3100000018.

[5] A. Merlin and H. Back. “Search for a Minimal loss operating spanning tree configuration in an urban power distribution system”. In: Proceedings of 5th Power System Computing Conference. 1975, pp. 1–18.

[6] Fred Glover, Gary Kochenberger, and Yu Du. “Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models”. In: 4OR 17.4 (2019), pp. 335–371. ISSN: 1614-2411. DOI: 10.1007/s10288-019-00424-y.
[7] Mark Lewis and Fred Glover. “Quadratic unconstrained binary optimization problem preprocessing: Theory and empirical analysis”. In: *Networks* 70.2 (2017), pp. 79–97. DOI: 10.1002/net.21751.

[8] Philipp Hauke et al. “Perspectives of quantum annealing: methods and implementations”. In: *Reports on Progress in Physics* 83.5 (May 2020), p. 054401. DOI: 10.1088/1361-6633/ab85b8.

[9] Rupak Biswas et al. “A NASA perspective on quantum computing: Opportunities and challenges”. In: *Parallel Computing* 64 (2017). High-End Computing for Next-Generation Scientific Discovery, pp. 81–98. ISSN: 0167-8191. DOI: 10.1016/j.parco.2016.11.002.

[10] Tameem Albash and Daniel A. Lidar. “Adiabatic quantum computation”. In: *Reviews of Modern Physics* 90.1 (Jan. 2018). ISSN: 1539-0756. DOI: 10.1103/revmodphys.90.015002.

[11] Catherine McGeoch and Pau Farré. *The D-Wave Advantage System: An Overview*. Technical Report 14-1049-A-A. D-Wave Systems Inc, 2020.

[12] *D-Wave Hybrid Solver Service: An Overview*. Whitepaper 14-1039-A-B. D-Wave Systems Inc, 2020.

[13] Andrew Lucas. “Ising formulations of many NP problems”. In: *Frontiers in Physics* 2 (2014), p. 5. ISSN: 2296-424X. DOI: 10.3389/fphy.2014.00005.

[14] Alex Fowler. “Improved QUBO Formulations for D-Wave Quantum Computing”. MA thesis. May 2017. DOI: 10.13140/RG.2.2.31829.73445.

[15] Reinhard Diestel. *Graph Theory*. 5th. Graduate Texts in Mathematics. Springer-Verlag GmbH Germany, 2017. ISBN: 978-3-662-53621-6. DOI: 10.1007/978-3-662-53622-3.

[16] Endre Boros and Peter L. Hammer. “Pseudo-Boolean optimization”. In: *Discrete Applied Mathematics* 123.1 (2002), pp. 155–225. ISSN: 0166-218X. DOI: 10.1016/S0166-218X(01)00341-9.

[17] D-Wave Systems Inc. *dwavebinarycsp Python package*. Accessed: 2021-09. URL: https://docs.ocean.dwavesys.com/en/stable/docs_binarycsp/sdk_index.html.

[18] Kelly Boothby et al. *Architectural considerations in the design of a third-generation superconducting quantum annealing processor*. 2021. arXiv: 2108.02322 [quant-ph].

[19] Jun Cai, William G. Macready, and Aidan Roy. *A practical heuristic for finding graph minors*. 2014. arXiv: 1406.2741 [quant-ph].

[20] Joseph Czyzyk, Michael P. Mesnier, and Jorge J. Moré. “The NEOS Server”. In: *IEEE Journal on Computational Science and Engineering* 5.3 (1998), pp. 68–75.

[21] Elizabeth D. Dolan. *The NEOS Server 4.0 Administrative Guide*. Technical Memorandum ANL/MCS-TM-250. Mathematics and Computer Science Division, Argonne National Laboratory, 2001.

[22] William Gropp and Jorge J. Moré. “Optimization Environments and the NEOS Server”. In: *Approximation Theory and Optimization*. Ed. by Martin D. Buhman and Arieh Iserles. Cambridge University Press, 1997, pp. 167–182.

[23] FICO Corp. *FICO® Xpress Optimization*. Accessed: 2022-03. URL: https://www.fico.com/en/products/fico-xpress-optimization.

[24] IBM Corp. *IBM CPLEX Optimizer*. Accessed: 2022-03. URL: https://www.ibm.com/analytics/cplex-optimizer.
[25] Gerald Gamrath et al. *The SCIP Optimization Suite 7.0*. Technical Report. Optimization Online, Mar. 2020. URL: http://www.optimization-online.org/DB_HTML/2020/03/7705.html.