electronics

MDPI

Article

Pattern QUBOs: Algorithmic Construction of 3SAT-to-QUBO Transformations

Sebastian Zielinski [1], Jonas Nüßlein [1], Jonas Stein [1], Thomas Gabor [1], Claudia Linnhoff-Popien [1] and Sebastian Feld [2,*]

[1] Institute for Informatics, LMU Munich, 80538 Munich, Germany; sebastian.zielinski@ifi.lmu.de (S.Z.); jonas.nuesslein@ifi.lmu.de (J.N.); jonas.stein@ifi.lmu.de (J.S.); thomas.gabor@ifi.lmu.de (T.G.); linnhoff@ifi.lmu.de (C.L.-P.)
[2] Quantum & Computer Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands
* Correspondence: s.feld@tudelft.nl

Abstract: One way of solving 3SAT instances on a quantum computer is to transform the 3SAT instances into instances of Quadratic Unconstrained Binary Optimizations (QUBOs), which can be used as an input for the QAOA algorithm on quantum gate systems or as an input for quantum annealers. This mapping is performed by a 3SAT-to-QUBO transformation. Recently, it has been shown that the choice of the 3SAT-to-QUBO transformation can significantly impact the solution quality of quantum annealing. It has been shown that the solution quality can vary up to an order of magnitude difference in the number of correct solutions received, depending solely on the 3SAT-to-QUBO transformation. An open question is: what causes these differences in the solution quality when solving 3SAT-instances with different 3SAT-to-QUBO transformations? To be able to conduct meaningful studies that assess the reasons for the differences in the performance, a larger number of different 3SAT-to-QUBO transformations would be needed. However, currently, there are only a few known 3SAT-to-QUBO transformations, and all of them were created manually by experts, who used time and clever reasoning to create these transformations. In this paper, we will solve this problem by proposing an algorithmic method that is able to create thousands of new and different 3SAT-to-QUBO transformations, and thus enables researchers to systematically study the reasons for the significant difference in the performance of different 3SAT-to-QUBO transformations. Our algorithmic method is an exhaustive search procedure that exploits properties of 4×4 dimensional *pattern QUBOs*, a concept which has been used implicitly in the creation of 3SAT-to-QUBO transformations before, but was never described explicitly. We will thus also formally and explicitly introduce the concept of *pattern QUBOs* in this paper.

Keywords: quantum annealing; pattern QUBO; automatic QUBO generation; QUBO; Ising; satisfiability; 3SAT; combinatorial optimization

Citation: Zielinski, S.; Nüßlein, J.; Stein, J.; Gabor, T.; Linnhoff-Popien, C.; Feld, S. Pattern QUBOs: Algorithmic Construction of 3SAT-to-QUBO Transformations. *Electronics* **2023**, *12*, 3492. https://doi.org/10.3390/electronics12163492

Academic Editor: João Soares

Received: 26 July 2023
Revised: 7 August 2023
Accepted: 15 August 2023
Published: 17 August 2023

1. Introduction

Satisfiability problems occupy a central place in computer science. They have been among the first problems for which NP-completeness has been shown [1], and they have often been used to proof the NP-completeness of other NP problems [2]. Informally, the satisfiablilty problem (SAT) of propositional logic is defined as follows: given a Boolean formula, is there any assignment of Boolean values to the involved variables such that the formula evaluates to "true"? [3]. These problems occur in various different domains, like planning [4], artificial intelligence [2], formal verification [5], automatic test-pattern generation [6], and more. Thus, in the past decades, several different methods for solving satisfiability problems have been developed [7–11]. Despite these efforts, to this day, no algorithm is known that can solve any instance of satisfiability problems in worst-case polynomial time.

Quantum computing, however, is a computational model that can solve some NP problems exponentially faster [12] than their classical counterparts by using phenomena of quantum mechanics for their calculations. There are two main approaches to quantum computing: the quantum gate model and adiabatic quantum computing. One method of solving hard problems with quantum computers is to transform them into an instance of a Quadratic Unconstrained Binary Optimization (QUBO), or into an instance of an Ising spin glass problem (Ising), because QUBO and Ising can be used as an input to quantum annealers [13] or the quantum approximate optimization algorithm (QAOA [14]) on quantum gate computers.

Due to recent advances in the manufacturing of quantum computers, the research community directs huge effort to employing quantum techniques to (NP-)hard problems in their respective fields. In the recent past, a particular research effort was to transform NP-hard optimization problems into instances of QUBO or Ising (e.g., see [15] for an overview). It is thus not surprising that the research community also studied satisfiability problems in the context of quantum computing (e.g., [16–20]).

To this day, several methods of transforming satisfiability problems to instances of QUBO have been published ([3,21,22], e.g.). These transformations result in different sizes or structures (i.e., the number, position and magnitude of the respective entries within the QUBO matrix) of the resulting QUBO instance. We will see shortly that these differences in the resulting QUBO instances can have a significant impact on the solution quality received by a quantum annealer. Finding new ways of transforming 3SAT instances into an instance of QUBO is still a very up-to-date topic, as just recently, new 3SAT-to-QUBO transformations were discovered and published [3]. All of these 3SAT-to-QUBO transformations probably have in common that they were carefully crafted manually by experts in this field using expertise, time and clever reasoning. Recent studies showed that the choice of a 3SAT-to-QUBO transformation can significantly impact the solution quality of quantum annealing ([11,23])—the results received by simply using another 3SAT-to-QUBO transformation for the same 3SAT instances vary up to an order of magnitude with respect to the number of correct answers returned by the quantum annealer. We thus argue that creating 3SAT-to-QUBO transformations solely by hand does not seem to be optimal. The space of potential 3SAT-to-QUBO transformations might be large (although we do not know how large, exactly), and the number of 3SAT-to-QUBO transformations in that potentially large space that can be discovered by an expert using clever reasoning is very limited.

Another recent research focus is to develop methods to automatically map given problems (i.e., objective functions) to instances of QUBO or Ising [24–27]. There are two major goals that these research efforts are trying to achieve. Firstly, by using computational methods to find problem-to-QUBO mappings automatically, one may be able to identify better (with respect to the solution quality received on quantum annealers) QUBO formulations for given problem classes. Secondly, these methods reduce the barrier to entry to quantum technology, as an instance of a given problem can be mapped automatically to an instance of QUBO or Ising (the input format for quantum annealing, i.e., the QAOA algorithm on quantum gate systems). Thus, there is no longer a need to understand or implement various mappings from a given problem instance to an instance of QUBO.

In this paper, as our first contribution, we thus propose an algorithmic method that is able to create thousands of new 3SAT-to-QUBO transformations automatically. All of the resulting 3SAT-to-QUBO transformations map 3SAT instances to instances of QUBO, which are of the dimension $n + m$, where n is the number of variables in a 3SAT instance and m is the number of clauses in a 3SAT instance. Firstly, this increases the number of known 3SAT-to-QUBO transformations from a few to several thousand. Secondly, because this method increased the number of known 3SAT-to-QUBO transformations from a few to several thousand, this method enables researchers to systematically study why some 3SAT-to-QUBO transformations lead to better solutions then other 3SAT-to-QUBO transformations

when solving 3SAT instances on a quantum annealer (which might ultimately lead to even better 3SAT-to-QUBO transformations).

As our second contribution, we formally introduce the concept of *pattern QUBOs*, which are a core part of our algorithmic method. The concept of pattern QUBOs has been previously used implicitly in the creation of 3SAT-to-QUBO transformations proposed by, e.g., Chancellor [22] or Nüßlein [3], but was never explicitly described before.

The remainder of this paper is structured as follows: Section 2 introduces necessary foundations for this paper, namely satisfiability problems, QUBO and Ising, and the maximum-weight independent set problem. In Section 3, we review state-of-the-art 3SAT-to-QUBO transformations that will be used in Section 5 in a case study. In Section 4, we formally introduce pattern QUBOs and explain this concept in depth. In Section 5, we present our algorithmic method to finding new 3SAT-to-QUBO transformations automatically. Additionally, we conduct a small benchmark study on D-Wave's quantum annealer Advantage_system4.1, in which we show that our proposed algorithmic method can create 3SAT-to-QUBO transformations that perform better than state-of-the-art 3SAT-to-QUBO transformations. In Section 6, we discuss the implications of our proposed method on the application of quantum annealing (as a SAT-solver). Furthermore, we show new research opportunities that are enabled by our algorithmic method. Finally, we conclude the paper in Section 7.

2. Background

2.1. Satisfiability Problems

Satisfiability problems are concerned with solving special types of Boolean formulae. Thus, we first introduce Boolean formulae as presented in [2].

Definition 1 (Boolean formula)**.** *Let $x_1, ..., x_n \in \{0, 1\}$ be Boolean variables. We set $0 := False$ and $1 := True$. A Boolean formula φ over n Boolean variables consists of the variables $x_1, ..., x_n$ and the logical operators* AND*($\wedge$),* OR*($\vee$),* NOT*($\neg$). For a given $z \in \{0, 1\}^n$ the expression $\varphi(z)$ denotes the value of φ when each of the Boolean variables x_i is assigned the value z_i. If there exists some assignment z such that $\varphi(z) = 1(True)$ we call φ satisfiable. Otherwise, we call φ unsatisfiable [2].*

In this work, we are addressing 3SAT problems, which are concerned with the satisfiability of Boolean formulae of a specific structure. The specific required structure is called the *Conjunctive Normal Form (CNF)*:

Definition 2 (Conjunctive Normal Form)**.** *Let φ be a Boolean formula over the Boolean variables $x_1, ..., x_n$. We say that φ is in CNF form, if φ is of the following form:*

$$\bigwedge_i \left(\bigvee_j l_{i_j} \right)$$

The terms l_{i_j} are called literals. The value of a literal is either a variable x_k or its negation $\neg x_k$ (for $k \in \{1, ..., n\}$). The terms $\left(\bigvee_j l_{i_j} \right)$ are called clauses. A kCNF *is a formula in CNF form in which all clauses contain at most k literals.*

We can now formally define 3SAT problems:

Definition 3 (3SAT)**.** *A* 3SAT *instance is a Boolean formula in 3CNF form, which comprises n Boolean variables and m clauses. The problem of deciding whether a* 3SAT *instance is satisfiable or not is the* 3SAT *problem.*

An example of a 3SAT instance is $\varphi_1(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$. In this case, φ_1 is satisfiable as, for example, $\varphi(1, 0, 0) = 1$.

Instances of the 3SAT problem, as well as many other NP-hard problems, are subject to the phase transition phenomenon [28]. Problem classes that are subject to the phase transition phenomenon can be easy to solve until their parameterization reaches a critical point from which on they are suddenly very hard to solve or even become unsolvable. For randomly created 3SAT instances with literals uniformly drawn from the variable pool of the 3SAT instance, the critical point of the phase transition is reached when the quotient of the number of clauses (m) and the number of variables (n) is approximately $\frac{m}{n} = 4.24$ [29].

2.2. Quadratic Unconstrained Binary Optimization (QUBO)

One way of solving problems on quantum computers is to transform them into an instance of a QUBO, as QUBOs are the accepted input for quantum algorithms like quantum annealing [13] and the QAOA algorithm [14] on quantum gate model computers.

Definition 4 (QUBO [30]). *Let $\mathcal{Q} \in \mathbb{R}^{n \times n}$ be a square matrix and let $x \in \mathbb{B}^n$ be an n-dimensional vector of Boolean variables. The QUBO problem is given as:*

$$\text{minimize} \quad H_{QUBO}(x) = x^T \mathcal{Q} x = \sum_{i}^{n} \mathcal{Q}_{ii} x_i + \sum_{i<j}^{n} \mathcal{Q}_{ij} x_i x_j$$

We call $H_{QUBO}(x)$ the (QUBO) energy of vector x. Matrix Q will also be called *QUBO matrix* or just *QUBO*. QUBO is closely related to the Ising spin glass problem (Ising), which is defined as follows:

$$\text{minimize} \quad H_{Ising}(s) = \sum_{i} h_i s_i + \sum_{i<j} J_{ij} s_i s_j \tag{1}$$

Here, h is an n-dimensional real-valued vector, J is a real-valued $n \times n$-dimensional upper triangular matrix, and $s_i \in \{-1, 1\}$ are spin variables.

QUBO and Ising are isomorphic. That means a QUBO instance can equivalently be expressed as an Ising instance and vice versa. To transform an instance of QUBO into an instance of Ising, one can use transformation $x_i = (s_i + 1)/2$ [31]. Because of this isomorphism, we will use QUBO and Ising interchangeably in this work. QUBO and Ising are both NP-hard problems [30].

2.3. Maximum Weight Independent Set (MWIS)

In this paper, we will use a 3SAT-to-QUBO transformation that requires an understanding of the Maximum Weight Independent Set (MWIS) problem. Thus, we will review the necessary foundations here. For the remainder of this section, let $G = (V, E)$ be an undirected graph with vertex set V and edge set E.

A subset $V' \subset V$ of the vertex set of G is called an *independent set*, if all vertices of V' are pairwise non-adjacent, i.e., if u, v are vertices in V', it follows that $(u, v) \notin E$ [32].

Definition 5 (Maximum Weight Independent Set). *In the MWIS problem, each vertex of G gets assigned a weight. Let $w_v \in \mathbb{R}^+$ be the weight that is assigned to vertex v. The MWIS problem is to find an independent set I of G, such that the total weight $w_{total} = \sum_{v \in I} w_v$ is the largest among all possible independent sets [32].*

To solve the MWIS problems on quantum annealers, they need to be transformed to QUBO instances. This can be performed as follows [21]:

If $Q_{ij} \geq |\min\{w_i, w_j\}|$ for all $(i, j) \in E$, then the maximum value of

$$H(x_1, ..., x_n) = -\sum_{i \in V} w_i x_i + \sum_{(i,j) \in E} Q_{ij} x_i x_j$$

is the total value w_{total} of the MWIS. The variables w_i denote the weight that was assigned to vertex i and the variables $x_i \in \{0,1\}$ are binary variables. In particular, the maximum weight independent set of graph G is given by $\{i \in V : x_i^* = 1\}$, where $(x_1^*, \ldots, x_n^*) = \arg\min_{(x_1,\ldots,x_n)} H(x_1, \ldots, x_n)$.

3. Related Work

In this section, we are going to review well-known 3SAT-to-QUBO transformations that will later be used as a baseline for our algorithmically generated 3SAT-to-QUBO transformations as well as the concept of pattern QUBOs.

3.1. Choi

The first 3SAT-to-QUBO transformation we are going to review was published by Choi [21]. In this approach, a 3SAT instance will be reduced to an instance of MWIS:

Given a 3SAT instance $\varphi(x_1, ..., x_n) = C_1 \wedge ... \wedge C_m$ with n variables and m clauses and an empty graph $G_{SAT} = (V_{G_{SAT}}, E_{G_{SAT}})$:

- For each clause $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$, we add a new 3-node clique to G_{SAT}. The three nodes are labeled by the literals $l_{i,1}, l_{i,2}$, and $l_{i,3}$ of clause C_i;
- A connection between nodes of different 3-node cliques will be added, if the labels of these nodes are in conflict. That is, if $l_{i,s} = \neg l_{j,t}$ for clause indices $i \neq j$ and $s, t \in \{1,2,3\}$, we add edge $(l_{i,s}, l_{j,t})$ to G_{SAT}.

In the resulting graph G_{SAT}, every vertex gets weighted by the same weight $w_{vertex} \in \mathbb{R}^+$, which can be chosen arbitrarily. This results in an MWIS instance in which the optimal solutions correspond to solutions of the input 3SAT instance. To solve this MWIS instance on a quantum annealer, it has to be transformed into an instance of QUBO, as described in Section 2.3.

3.2. Chancellor

Chancellor [22] proposed different methods to transform 3SAT instances to Ising spin glass problems. In this paper, we will use the first proposed method in the "special cases" section of [22]. The underlying idea of this transformation is that for each clause of a 3SAT instance, a new Ising spin glass problem will be constructed. All the created clause Ising spin glass problems will then be "added" together. We will now proceed to explain Chancellor's method as proposed in [22]:

Let $C_i = (x_1 \vee x_2 \vee x_3)$ be a clause of a 3SAT instance over the Boolean variables x_1, x_2, x_3. If we set 1 := True and 0 := False, then the following holds:

$$(x_1 \vee x_2 \vee x_3) = x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3 + x_1x_2x_3 \tag{2}$$

This expression is maximized (the value is 1), whenever the clause is satisfied, and zero otherwise. As we defined Ising (and QUBO) to be a minimization problem, we transform Equation (2) into the equivalent minimization version:

$$-(x_1 \vee x_2 \vee x_3) = -x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3 - x_1x_2x_3 \tag{3}$$

Now, whenever the clause is satisfied, the value is -1 and zero otherwise. As Chancellor's approach uses spin variables, the Boolean variables x_1, x_2, x_3 in Equation (3) will be replaced by corresponding spin variables $s_1, s_2, s_3 \in \{-1, 1\}$ by using transformation $x_i = \frac{1}{2}(s_i + 1)$. This leads to:

$$-(x_1 \vee x_2 \vee x_3) = -\frac{1}{8}s_1 - \frac{1}{8}s_2 - \frac{1}{8}s_3 + \frac{1}{8}s_1s_2 + +\frac{1}{8}s_1s_3 + \frac{1}{8}s_2s_3 - \frac{1}{8}s_1s_2s_3 - \frac{7}{8} \tag{4}$$

We will call the right hand side of Equation (4) the *spin representation of a clause*. The goal now is to create an Ising spin glass problem in which all the assignments to Boolean variables x_i (or equivalent spin variables s_i) that satisfy the clause have the exact same

minimal energy, while the one assignment that does not satisfy the clause has a higher energy. We observe that the rightmost term in Equation (4) that involves spin variables is cubic, but Ising and QUBO can only model quadratic interactions natively. Chancellor showed that this cubic term can nevertheless be expressed in Ising, by adding an additional spin s_a, also called *ancilla* spin. The cubic term is modeled as follows:

$$I_{cubic}(s_1, s_2, s_3) = J \sum_{i=1}^{3} \sum_{j=1}^{i-1} c(i)c(j)s_i s_j + h \sum_{i=1}^{3} c(i)s_i + J_a \sum_{i=1}^{3} c(i)s_i s_a + h_a s_a \tag{5}$$

The terms $c(i)$ represent the sign of the corresponding Boolean variable x_i in the clause we are trying to implement. As all the variables in the clause of Equation (4) are not negated, $c(i) = 1$ for all i. Parameters J, J_a, h, h_a need to satisfy the following constraints:

- h is the value of the coefficient of the cubic term in the spin representation of the clause;
- $h_a = 2h$;
- $J_a = 2J > |h|$, which means the magnitude of J can be chosen freely, as long as $2J > |h|$ holds.

This transformations thus has a free parameter, namely J, which needs to be chosen first, before using this transformation. By following these instructions, we have now constructed an Ising spin glass representation of the cubic term. To obtain to the Ising spin glass representation of the clause, we want to implement we add the coefficients of the linear and quadratic terms of the spin representation of the clause we are implementing (see Equation (4)) and the coefficients of the linear and quadratic terms of the Ising representation of the cubic term (see Equation (5)) together:

$$I_{clause} = I_{cubic} + l_1 s_1 + l_2 s_2 + l_3 s_3 + q_{12} s_1 s_2 + q_{13} s_1 s_3 + q_{23} s_2 s_3 \tag{6}$$

The terms $l_i, q_{ij} \in \{+\frac{1}{8}, -\frac{1}{8}\}$ are the coefficients of the corresponding linear and quadratic terms of Equation (4).

This procedure is repeated for all clauses of the given 3SAT instance. Finally, we want to construct the Ising representation of the 3SAT instance by "combining" all of the Ising representations of the clauses. That is, we are going to superimpose the construction of the individual clauses on a common set of logical spin variables. To do so, suppose a given 3SAT formula consists of variables $x_1, ..., x_n$ and m clauses $C_1, ..., C_m$. The corresponding Ising minimization problems will consist of spin variables $s_1, ..., s_n$, which directly correspond to Boolean variables $x_1, ..., x_n$ and m additional ancilla qubits $s_{a_1}, ..., s_{a_m}$, which are needed to model the cubic term of each clause (see Equation (5)).

Superimposing works as follows [22]:

- For each s_i: Let K be the set of all clauses in which x_i or $-x_i$ appears as a variable. Then, $h_i = \sum_{l=1}^{|K|} h(i)_l$. The terms $h(i)_l$ are calculated by $h(i)_l = h \cdot c(i)_l + coeff(s_i)_l$. The term $h \cdot c(i)_l$, where $c(i)_l$ is the sign of variable x_i in the l-th clause of K, results from Equation (5). The term $coeff(s_i)_l$ is the coefficient of the spin variable s_i in the spin representation of the l-th clause of K;
- For all pairs of spins s_i, s_j where $i < j$: Let P be the set of clauses in which variables x_i and x_j appear together. Then, the values of the upper triangular matrix J_{ij} are given by $J_{ij} = \sum_{l=1}^{|P|} J(ij)_l$. The terms $J(ij)_l$ are calculated by $J(ij)_l = J \cdot c(i)_l c(j)_l + coeff(s_i s_j)_l$. The term $coeff(s_i s_j)_l$ is the coefficient of the quadratic term $s_i s_j$ in the spin representation of the l-th clause of P;
- All quadratic and linear values that include an ancilla qubit will not be modified at all.

As Ising and QUBO are isomorphic, the resulting Ising representation of a given 3SAT instance can be transformed into an equivalent QUBO instance. As the remainder of this work will only use QUBO instances, we will refer to Chancellor's approach as a 3SAT-to-QUBO method, by implicitly using the isomorphism between Ising and QUBO.

As the mechanism of superimposing will be heavily used in the remainder of this paper, we illustrate Chancellor's approach and the method of superimposing in a simple example.

Example 1. *Suppose we are given* 3SAT *instance* $\varphi(x_1, x_2, x_3, x_4) = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_4)$. *For each of the clauses, we construct an Ising problem that corresponds to the clause, as explained previously. To do so, we use* $J = |h| = \frac{1}{8}$. *We then transform the resulting clause Ising problems into an instance of QUBO. The resulting QUBOs are shown in Table 1.*

Table 1. QUBOs for 3SAT clauses $(x_1 \vee x_2 \vee x_3)$ and $(x_1 \vee \neg x_2 \vee x_4)$.

(**a**) QUBO for the clause $(x_1 \vee x_2 \vee x_3)$				
	x_1	x_2	x_3	A_1
x_1	-2	1	1	1
x_2		-2	1	1
x_3			-2	1
A_1				-2
(**b**) QUBO for the clause $(x_1 \vee \neg x_2 \vee x_4)$				
	x_1	x_2	x_4	A_2
x_1	-1	0	1	1
x_2		0	0	1
x_4			-1	1
A_1				-1

The variables A_1 and A_2 in Table 1 are the ancilla variables for each of the clauses. Note that, although variable x_2 is negated in clause $(x_1 \vee \neg x_2 \vee x_4)$, we use x_2 as the QUBO variable in the resulting clause QUBO (shown in Table 1b). By following the process of superimposing, one receives the QUBO shown in Table 2. We colored the entries of the QUBOs shown in Table 2, so one can easily understand the process of superimposing visually. All orange-colored values in Table 2 are taken from the QUBO for the clause $(x_1 \vee x_2 \vee x_3)$ (shown in Table 1a), and all the black values are taken from the QUBO for the clause $(x_1 \vee \neg x_2 \vee x_4)$ (shown in Table 1b).

Table 2. QUBOs for 3SAT clauses $(x_1 \vee x_2 \vee x_3)$ and $(x_1 \vee \neg x_2 \vee x_4)$.

	x_1	x_2	x_3	x_4	A_1	A_2
x_1	$-2-1$	$1+0$	1	1	1	1
x_2		$-2+0$	1	0	1	1
x_3			-2		1	
x_4				-1		1
A_1					-2	
A_2						-1

4. Pattern QUBOs

The core contribution of our paper, namely the algorithmic method, which we will propose in Section 5, is based on exploitation of the concept of *pattern QUBOs*. As our first contribution, we will now formally introduce the concept of pattern QUBOs, which have been implicitly used in the creation of known 3SAT-to-QUBO transformations, like the 3SAT-to-QUBO transformations by Chancellor [22] or Nüßlein [3], but never was explained explicitly before.

4.1. QUBO Energy of a 3SAT *Assignment*

We start by examining the QUBO for clause $(x_1 \vee x_2 \vee x_3)$ (shown in Table 1a). We call this QUBO Q_0 in the remainder of this section. QUBO Q_0 can be used to find satisfying assignments for clause $(x_1 \vee x_2 \vee x_3)$. That is, each Boolean vector $\mathbf{x} = (x_1, x_2, x_3, a_1)$ that minimizes $\mathbf{x}^T Q_0 \mathbf{x}$ corresponds to a satisfying assignment of clause $(x_1 \vee x_2 \vee x_3)$. As a_1 is an ancilla variable that is not part of the set of variables of a given clause, for each assignment of Boolean values to variables x_1, x_2, x_3, there are two corresponding Boolean vectors in the QUBO minimization $\mathbf{x}^T Q_0 \mathbf{x}$: $(x_1, x_2, x_3, a_1 = 0)$ and $(x_1, x_2, x_3, a_1 = 1)$. We now define the energy of an assignment of Boolean values to the variables of a 3SAT instance:

Definition 6 (Energy of an assignment)**.** *Let $x_1, ..., x_n$ be the variables of a* 3SAT *instance and let Q be a QUBO instance consisting of the variables $x_1, ..., x_n$ as well as m additional ancilla variables $a_1, ..., a_m$ to represent the given* 3SAT *instance as a QUBO instance. Given an assignment $x_1 = k_1, ..., x_n = k_n$, where $k_1, ..., k_n \in \{0, 1\}$, of Boolean values to variables $x_1, ..., x_n$, we define a column vector $\boldsymbol{x}$ as $\boldsymbol{x} = (k_1, ..., k_n, a_1, ..., a_m)$. The energy of an assignment of Boolean values to the variables $x_1, ..., x_n$ is given by:*

$$\mathcal{E}(x_1 = k_1, ..., x_n = k_n) = \min \boldsymbol{x}^T Q \boldsymbol{x} \tag{7}$$

Additionally, if QUBO Q was used to calculate the energy of an assignment $x_1 = k_1, ..., x_n = k_n$, we will also say: "The assignment $x_1 = k_1, ..., x_n = k_n$ has energy $\mathcal{E}$ in Q". With this, we emphasize that the specific QUBO Q was used to calculate the energy for assignment $x_1 = k_1, ..., x_n = k_n$.

The energy of an assignment is thus the value one obtains by choosing the values of the m ancilla variables $a_1, ..., a_m \in \{0, 1\}^m$ such that the corresponding QUBO optimization process yields the lowest value.

As an example, we calculate the energy for the assignment $x_1 = x_2 = x_3 = 0$ by using clause QUBO Q_0. To do so, we first define two column vectors v_1 and v_2:

- Vector $v_1 := (0, 0, 0, a_1 = 0)^T$;
- Vector $v_2 := (0, 0, 0, a_1 = 1)^T$.

The energy of the assignment $x_1 = x_2 = x_3 = 0$ is thus given by:

$$\mathcal{E}(0, 0, 0) = \min(v_1^T Q_0 v_1, v_2^T Q_0 v_2) = \min(0, -1) = -1$$

Hence, the assignment $x_1 = x_2 = x_3 = 0$ has energy -1 in Q.

Next, we define the term *clause QUBO* that will heavily be used in the following sections.

Definition 7 (Clause QUBO)**.** *Let C be a clause of a* 3SAT *instance. Let S_{SAT} be the set of assignments of Boolean values to the variables of clause C that satisfy clause C. A* clause QUBO *is a QUBO for which each $s \in S_{SAT}$ has the same minimal energy and for which the only assignment that does not satisfy clause C has a higher energy than any of the assignments of S_{SAT}.*

4.2. Pattern QUBOs

To introduce the concept of pattern QUBOs, we want to first point out the following (implicitly already used) observation. Suppose we are given the 3SAT instance of Example 1, $\varphi(x_1, x_2, x_3, x_4) = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_4)$. To calculate the energy of an assignment of Boolean values to variables $x_1, ..., x_4$, there are two equivalent ways:

(i) Calculate the energy of $x_1 = 1, x_2 = 0, x_3 = 0$ for the first clause by using the QUBO shown in Table 1a. Calculate the energy of $x_1 = 1, x_2 = 0, x_4 = 0$ for the second clause by using the QUBO shown in Table 1b. Add both values;

(ii) Calculate the energy of $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 0$ by using the superimposed QUBO shown in Table 2.

It can easily be verified that this is true for all possible assignments (i.e., by visually understanding the construction of the superimposed QUBO in Table 2). Thus, we note that

for each assignment, the energy of the superimposed QUBO (shown in Table 2) is equal to the sum of the energies of the respective assignments for the smaller QUBOs (shown in Table 1) that were used to create the superimposed QUBO.

An important consequence of this observation is that any assignment that does not satisfy the 3SAT instance cannot possess the lowest energy value that is possible for the superimposed QUBO. That is the case because the QUBOs (shown in Table 1) that were used to create the superimposed QUBO (shown in Table 2) are clause QUBOs (see Definition 7)—i.e., they were constructed such that only satisfying assignments for that clause have the lowest possible energy. An assignment that does not satisfy the 3SAT instance also does not satisfy at least one clause. This assignment has a higher than optimal energy for this clause. As the energy of an assignment of the superimposed QUBO is the sum of the energies of the respective assignments for the smaller QUBOs that were used to create the superimposed QUBO, it follows that the energy for this non-satisfying assignment cannot be the lowest possible energy of the superimposed QUBO. Thus, as long as every clause is transformed into a corresponding clause QUBO (see Definition 7), one can create a QUBO representation of a 3SAT instance by combining all the clause QUBOs via superimposing.

For 3SAT problems specifically, we observe that each clause of a 3SAT instance has either exactly zero, one, two, or three negated variables. By rearranging the clause, we can assume, without loss of generality, that the negated variables are always at the end of the clause. Thus, each 3SAT clause is represented by one of the following four types of clauses [3]:

- Type 0 := $(a \vee b \vee c)$;
- Type 1 := $(a \vee b \vee \neg c)$;
- Type 2 := $(a \vee \neg b \vee \neg c)$;
- Type 3 := $(\neg a \vee \neg b \vee \neg c)$.

By using the previous argument, to create a new 3SAT-to-QUBO transformation, we only need to find one clause QUBO for each of the four clause types. We will use the term *pattern QUBOs* for QUBOs that are clause QUBOs for one of the four types (type 0–3) of clauses. We now demonstrate why the term *pattern QUBO* is justified.

A pattern QUBO for a type 0 clause is shown in Table 3.

Table 3. Pattern QUBO for type 0 clause $(a \vee b \vee c)$.

	a	b	c	A
a	-2	1	1	1
b		-2	1	1
c			-2	1
A				-2

If we want to transform a given 3SAT instance with m clauses to a QUBO instance, and we encounter a clause with zero negations, say $(x_1 \vee x_3 \vee x_5)$, we can substitute the variables a, b, c in the type 0 pattern QUBO (shown in Table 3) by x_1, x_3, x_5. This substitution of variables leads to a clause QUBO (see Definition 7) for the clause $(x_1 \vee x_3 \vee x_5)$. By repeating this process for all m clauses (using the correct type 0–3 pattern QUBO for the respective clauses), we create m QUBO representations of individual 3SAT clauses. As explained earlier, the resulting clause QUBOs can then be combined into a single QUBO that represents the whole 3SAT instance, by using the method of superimposing.

A certain 3SAT-to-QUBO transformation can thus be seen as an ordered tuple (c_0, c_1, c_2, c_3), where c_i is a pattern QUBO for clause type i (for $0 \leq i \leq 3$). Note in particular that there exist multiple such tuples, which we will demonstrate in the coming section.

5. Algorithmic Method

In this section, we are going to present an algorithmic method that uses the concept of pattern QUBOs to create thousands of new 3SAT-to-QUBO transformations automatically. We will also perform a case study, in which we solve 3SAT instances using both state-of-the-art and created-by-algorithm 3SAT-to-QUBO transformations on D-Wave's quantum annealer Advantage_system4.1. A python implementation of the proposed framework and all the 3SAT-instances we used in the case study are provided in the GitHub repository associated with this paper [33].

5.1. Algorithm Description

As explained in Section 4.2, to create new 3SAT-to-QUBO transformations, it suffices to find pattern QUBOs for type 0–3 clauses. These pattern QUBOs can be reused and combined (by superimposing) to create a QUBO instance corresponding to a given 3SAT instance.

We will now present an algorithmic method (see Algorithm 1) that uses the blueprint for a pattern QUBO shown in Table 4 to create new pattern QUBOs for clause types 0–3.

Table 4. Blueprint for a pattern QUBO of any clause type.

	a	***b***	***c***	***A***
a	Q_1	Q_2	Q_3	Q_4
b		Q_5	Q_6	Q_7
c			Q_8	Q_9
A				Q_{10}

This method has to find values for all Q_i, such that the resulting QUBO is a clause QUBO (see Definition 7) for the given clause type. As an input to this method, we have to specify a set of values from which Q_i can be chosen. If this set is small enough, we can use exhaustive search to search all possible combinations of the 10 variables:

Algorithm 1 Search procedure to generate pattern QUBOs for type 0–3 clauses automatically

Require: Minimum value $min \in \mathbb{R}$, maximum value, $max \in \mathbb{R}$, and step size $step \in \mathbb{R}$
Require: Upper triangular matrix Q. All values of Q (Q_1 to Q_{10}) should initially be min.
procedure SEARCH CLAUSE QUBOS(clause type)
FoundQUBOS = {}
while $Q_i <= max \ \ \forall Q_i$ **do**
if IsClauseQUBO(Q) **then**
FoundQUBOS.insert(Q)
end if
$Q_1 \leftarrow Q_1 + step$
for $i = 1$ to 9 **do**
if $Q_i > max$ **then**
$Q_i \leftarrow min$
$Q_{i+1} \leftarrow Q_{i+1} + step$
else
break
end if
end for
end while
return FoundQUBOS
end procedure

The function *IsClauseQUBO(Q)* in line 4 checks whether the modified upper triangular matrix Q is a clause QUBO for the given clause type. That is, this function checks whether

all satisfying assignments for that clause type have the same energy in Q, while the one assignment that does not satisfy the clause specified by the given clause type has a higher energy than any of the satisfying assignments.

By using this method and $min = -1$, $max = 1$ and $step = 1$, we find six pattern QUBOs for clause type 0, seven pattern QUBOs for clause type 1, six pattern QUBOs for clause type 2, and 8 pattern QUBOs for clause type 3. As any tuple (c_0, c_1, c_2, c_3), where c_i is a pattern QUBO for clause type i ($0 \leq i \leq 3$), can be seen as a valid 3SAT-to-QUBO transformation, we have created a total of $2016 = 6 \cdot 7 \cdot 6 \cdot 8$ 3SAT-to-QUBO transformations. By decreasing min and increasing max, even more combinations will be found.

Because we are using a new ancilla variable in each of the pattern QUBOs, all 3SAT-to-QUBO transformations created by this method result in QUBO matrices of dimensions $n + m$, where n is the number of variables and m is the number of clauses of a 3SAT instance. Thus, by choosing appropriate values for min, max, and $step$, this method can also produce previously published 3SAT-to-QUBO transformations of size $n + m$. Chancellor's 3SAT-to-QUBO transformation, for example, can be found with our method by using $min = -2$, $max = 1$, and $step = 1$.

5.2. Case Study Using Quantum Hardware

We are now going to conduct a case study in which we show that our method can create competitive 3SAT-to-QUBO transformations by comparing the results of a 3SAT-to-QUBO transformation created by our algorithm with the results of well known state-of-the-art 3SAT-to-QUBO transformations when solving 3SAT instances on D-Wave's quantum annealer *Advantage_system4.1*. That is, we will show that our automatically created 3SAT-to-QUBO transformations can solve equally many 3SAT instances as state-of-the-art 3SAT-to-QUBO transformations, and that they can find correct solutions with an equal or higher probability than state-of-the-art 3SAT-to-QUBO transformations.

To do so, we create 1000 3SAT instances according to the following description:

- Each 3SAT instance has $m = 50$ clauses and $n = 12$ variables ($m/n \approx 4.16$);
- For each of the m clauses, we uniformly select three different variables from the n variables of the instance. Additionally, there is a 50% chance that each of the three chosen variables will get negated in that clause;
- Verify that the created 3SAT instance is solvable (using a SAT solver).

As explained in Section 5.1, by using our method with parameters $min = -1$, $max = 1$, and $step = 1$, we can create 2016 3SAT-to-QUBO transformations. As we cannot evaluate all of these transformations on real quantum hardware in a meaningful study (for cost reasons), we solved one of the 1000 created 3SAT instances with all of the 2016 different 3SAT-to-QUBO transformations on classical hardware using a tabu search. Finally, out of all 3SAT-to-QUBO transformations that were performing well, we chose on at random. We call this transformation the *Algorithm QUBO* in the following evaluation.

We now apply Chancellor's, Choi's and the Algorithm's 3SAT-to-QUBO transformation to all of the formerly created 1000 3SAT instances. Each of the resulting 3000 QUBO instances will be solved 1000 times on D-Wave's quantum annealer *Advantage_system4.1*. We used default parameters to solve all the QUBO instances on the quantum annealer. Specifically, we used the sample method of D-Waves *Embedding Composite* class (https://docs.ocean.dwavesys.com/projects/system/en/stable/reference/generated/dwave.system.composites.EmbeddingComposite.sample.html#dwave.system.composites.EmbeddingComposite.sample, accessed on 14 August 2023) without any modifications and "num_reads=1000" as the only input to the "**parameters" dictionary. The results of this experiment are shown in Table 5.

Table 5. Results of the case study. A total of 1000 3SAT instances were solved 1000 times by Chancellor's, Choi's, and the Algorithm's 3SAT-to-QUBO transformation.

QUBO	#Solved Instances	#Correct Solutions
Algorithm QUBO	963 (96.3%)	181,614 ($\approx$ 18.16%)
Chancellor	958 (95.8%)	139,168 ($\approx$ 13.91%)
Choi	792 (79.2%)	45,255 ($\approx$ 4.53%)

We can see that the Algorithm's 3SAT-to-QUBO transformation solves the most of the 1000 3SAT instances, closely followed by Chancellor's 3SAT-to-QUBO transformation. However, the Algorithm's QUBOs finds approximately 30% more correct solutions than Chancellor's QUBOs. We also observe that the results of Choi's QUBOs are far behind the results of Chancellor's and the Algorithm's QUBOs. The latter observation is in accordance with the comparison of these two transformations in [23], where it was also observed that Choi's QUBOs seem to perform significantly worse than Chancellor's QUBOs on similar 3SAT instances.

The results of this case study show that our algorithmic method can produce 3SAT-to-QUBO transformations that can compete with or even outperform state-of-the-art 3SAT-to-QUBO transformations with regards to the number of solved instances, i.e., the number of correct solutions. We note that in the benchmark study performed in [23], the worst-performing 3SAT-to-QUBO transformation produced QUBO instances of size $n + m$. As our algorithmic method also creates $n + m$ sized 3SAT-to-QUBO transformations, the worst-performing 3SAT-to-QUBO transformation of the benchmark study in [23] can also be found with our algorithmic method. Thus, it should be clear that not every 3SAT-to-QUBO transformation created by our proposed algorithmic method will outperform state-of-the-art 3SAT-to-QUBO transformations.

6. Discussion

A recent benchmark study (see [23]) showed that the choice of a 3SAT-to-QUBO transformation can significantly impact the solution quality when solving 3SAT instances via quantum annealing. The results varied up to an order of magnitude with respect to to the received solution quality (i.e., number of correct answers to a given 3SAT instance returned by the quantum annealer), depending solely on the 3SAT-to-QUBO transformation used. This result has an important consequence for the evaluation of the problem solving capabilities of quantum annealing with regards to 3SAT instances.

Algorithms, or more generally, problem solvers are naturally classified according to their ability to solve a given problem. Important metrics are the time it takes the solver to reach a correct (or good enough) answer, as well as the capability to find correct (or good enough) answers in the first place. Now that quantum computers (and in our situation, quantum annealers especially) became a reality, it is natural to compare the ability of quantum annealers to solve 3SAT problems against already existing classical methods. Leaving aside obvious problems of current quantum computers, like missing error-correction or limited amount of qubits (and thus limited size of problems that can be solved), it is still the case that we are comparing highly optimized classical methods vs. newly available and less optimized quantum hardware and quantum methods.

In classical computing, we have been dedicating decades to the development of different algorithms that are able to solve 3SAT instances or heuristics that produce very good answers (although not satisfying solutions). As part of these efforts, researchers explored the applicability of many different problem-solving techniques from various different domains to the 3SAT problem, which ultimately led to the availability of powerful SAT solvers today. In the relatively new field of quantum computing, we are just learning how to properly use currently available quantum solving techniques. In quantum annealing, the algorithm is quite fixed; however, the underlying problem representation can be varied by using different 3SAT-to-QUBO transformations. Thus, analogously to the classical case where one tries to improve the problem

solving capabilities of classical hardware by finding more powerful algorithms, finding better 3SAT-to-QUBO transformations can be interpreted as an effort to improve the problem solving capabilities of quantum hardware where the algorithm is fixed. This is especially true in the current NISQ era. It may be the case that different solution landscapes created by the use of different QUBO representations of a given problem can deal better (or worse) with the given impact of noise on the process of quantum annealing. Of course, the performance gain of changing the QUBO representation of a 3SAT instance is limited by the general problem solving capability of the underlying algorithm (quantum annealing, in this case). In other words, by assuming that future quantum computers improve in their general problem solving capabilities (i.e., by reducing the influence of noise), the goal of finding better QUBO transformations for given problems is to enhance the problem-solving capabilities of the quantum computer even further (i.e., further than just considering technical improvements) and thus making quantum computers a viable or even superior (with respect to classical methods) solver for certain problems in the future.

Despite the importance of the 3SAT-to-QUBO transformation on the solution quality, only a few 3SAT-to-QUBO transformations are known. This is what we have addressed in this paper. Our algorithmic method, although simple and certainly limited in the scalability (as exhaustive search scales exponentially in this case), is able to create thousands of new 3SAT-to-QUBO transformations. This has an immediate consequence for the application of solving 3SAT instances via quantum annealing: it is now possible to choose from a significantly wider variety of 3SAT-to-QUBO transformations. As we increased the number of available 3SAT-to-QUBO transformations by orders of magnitude, it is very likely that some of the new 3SAT-to-QUBO transformations created by our algorithmic method will perform significantly better than currently known 3SAT-to-QUBO transformations. We provided some empirical evidence for this hypothesis by performing a small case study in Section 5, where we observed that one of the 3SAT-to-QUBO transformations created by our algorithmic method lead to approx 30% (i.e., approx 400%) more correct answers than current state-of-the-art 3SAT-to-QUBO transformations.

Another implication of our algorithmic method is that it increases the accessibility to quantum technology. Previously, it was necessary to understand how exactly one has to transform a given 3SAT instance into an instance of QUBO. This process can be quite complicated (as seen for example in the transformation of Chancellor described in Section 3.2). Because of the effort that needs to be put into understanding a single transformation, it is plausible that someone who just wants to apply quantum annealing as a solver for a 3SAT problem might just try to understand and implement a single (or a few) 3SAT-to-QUBO transformations and evaluate the quantum problem solving capabilities based on the use of the very limited amount of 3SAT-to-QUBO transformations. As previously stated, this may result in an impression that does not reflect the currently possible problem solving capabilities of quantum annealing correctly (i.e., the results may be orders of magnitude worse, because a sub-optimal 3SAT-to-QUBO transformation has been used). By understanding the concept of pattern QUBOs, which we introduced in Section 4, it is possible to create a large number of different 3SAT-to-QUBO transformations with our algorithmic method, including currently known 3SAT-to-QUBO transformations like Chancellor's transformation, without having to understand the logic behind each and every one of these transformations. From an application perspective, our algorithmic method thus replaces the need for understanding of a lot of different concepts behind a lot of different 3SAT-to-QUBO transformations by the understanding of a single concept—the concept of pattern QUBOs.

Because there are now thousands of 3SAT-to-QUBO transformations available, researchers can now systematically study why some 3SAT-to-QUBO transformations perform significantly better (or worse) than others. Results of these investigations might lead to new insights that lead to the creation of even better-performing 3SAT-to-QUBO transformations. Finally, we want to discuss some additional ideas for future research. Although our algorithmic methods increased the number of known 3SAT-to-QUBO transformations by orders of magnitude,

it has a downside when it comes to exploring the space of all possible 3SAT-to-QUBO transformations: the algorithm scales exponentially. Thus, although our algorithm enables to explore a significantly larger space of possible 3SAT-to-QUBO transformations, it is possible that we still can explore only a small part of the space of all possible 3SAT-to-QUBO transformations. When finding methods of creating pattern QUBOs more efficiently, one can explore an even bigger part of the space of possible 3SAT-to-QUBO transformations. This may possibly be performed by employing methods from evolutionary computing.

By solving large amounts of different 3SAT instances with a given number of 3SAT-to-QUBO transformations, it may be possible to use techniques from machine learning to create some form of recommendation system that suggests a 3SAT-to-QUBO transformation when providing a 3SAT input.

The efficiency of quantum annealing depends on the proper selection of the annealing schedule, i.e., the schedule that describes how the perturbations decrease with time [34]. A recent publication ([35]) proposes an optimization of annealing schedules via machine learning. Thus, a promising idea for the future is to extend our proposed framework by also including techniques to automatically find better (than the default) annealing schedules for given QUBOs. We suspect that an optimization of annealing schedules might be necessary for each instance individually, or at least for each class of structurally similar instances.

7. Conclusions

In this paper, we presented an algorithmic method that enables the automatic creation of new 3SAT-to-QUBO transformations. With this method, we increased the number of known 3SAT-to-QUBO transformations from approximately a dozen to several thousand. Our algorithmic method builds upon the concept of *pattern QUBOs*, which can be seen as a generalization of techniques that have been previously used in the creation of 3SAT-to-QUBO transformations. By conducting a small benchmark study, we have shown, that our algorithmic method can produce 3SAT-to-QUBO transformations that perform better than state-of-the-art 3SAT-to-QUBO transformations. Because our method is able to create thousands of new 3SAT-to-QUBO transformations, researchers can now study the differences between well- and badly-performing 3SAT-to-QUBO transformations. Results of this investigation might lead to insights that help to create even better 3SAT-to-QUBO transformations. Finally our algorithmic method also increases the accessibility of quantum technologies. Previously, it was necessary to understand the often complicated logic behind a given 3SAT-to-QUBO transformation in order to be able to apply quantum annealing as solver for 3SAT instances. However, as it was shown, that the choice of a 3SAT-to-QUBO can significantly impact the solution quality of quantum annealing, it is also necessary to understand and use different 3SAT-to-QUBO transformations to solve given 3SAT instances. By using our method, one only needs to understand a single concept (pattern QUBOs) to be able to use thousands of different 3SAT-to-QUBO transformations, including current state-of-the-art 3SAT-to-QUBO transformations, without the need to understand their (potentially) complicated underlying logic. Thus, in addition to seeing pattern QUBOs as a generalization of known concepts, our algorithmic method can also be seen as an abstraction layer for the application of 3SAT-to-QUBO transformations.

Author Contributions: Conceptualization, S.Z. and J.N.; methodology, S.Z.; software, S.Z.; validation, S.Z., J.N. and J.S.; investigation, S.Z.; writing—original draft preparation, S.Z.; writing—review and editing, J.N., J.S., S.F., T.G. and C.L.-P.; supervision, T.G., S.F. and C.L.-P.; funding acquisition, C.L-P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the German Federal Ministry of Education and Research through the funding program "quantum technologies—from basic research to market" (contract number: 13N16196).

Data Availability Statement: Used 3SAT instances and the code for the proposed algorithm and framework can be found in the git repository corresponding to this paper [33].

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Cook, S.A. The complexity of theorem-proving procedures. In Proceedings of the Third Annual ACM Symposium on Theory of Computing, Shaker Heights, OH, USA, 3–5 May 1971; pp. 151–158.
2. Arora, S.; Barak, B. *Computational Complexity: A Modern Approach*; Cambridge University Press: Cambridge, MA, USA, 2009.
3. Nüßlein, J.; Zielinski, S.; Gabor, T.; Linnhoff-Popien, C.; Feld, S. Solving (Max) 3-SAT via Quadratic Unconstrained Binary Optimization. *arXiv* **2023**, arXiv:2302.03536.
4. Kautz, H.A.; Selman, B. Planning as Satisfiability. In Proceedings of the ECAI, Vienna, Austria, 3–7 August 1992; Volume 92, pp. 359–363.
5. Prasad, M.R.; Biere, A.; Gupta, A. A survey of recent advances in SAT-based formal verification. *Int. J. Softw. Tools Technol. Transf.* **2005**, *7*, 156–173. [CrossRef]
6. Marques-Silva, J. Practical applications of boolean satisfiability. In Proceedings of the 2008 9th International Workshop on Discrete Event Systems, Goteborg, Sweden, 28–30 May 2008; pp. 74–80.
7. Schoning, T. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In Proceedings of the 40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039), New York, NY, USA, 17–18 October 1999; pp. 410–414.
8. Selman, B.; Kautz, H.A.; Cohen, B. Local search strategies for satisfiability testing. *Cliques Color. Satisf.* **1993**, *26*, 521–532.
9. Marques Silva, J.P.; Sakallah, K.A. *GRASP—A New Search Algorithm for Satisfiability*; Springer: Berlin, Germany, 2003.
10. Davis, M.; Logemann, G.; Loveland, D. A machine program for theorem-proving. *Commun. ACM* **1962**, *5*, 394–397. [CrossRef]
11. Kurin, V.; Godil, S.; Whiteson, S.; Catanzaro, B. Improving SAT solver heuristics with graph networks and reinforcement learning. *arXiv* **2019**, arXiv:1909.11830.
12. Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134.
13. McGeoch, C.C. Theory versus practice in annealing-based quantum computing. *Theor. Comput. Sci.* **2020**, *816*, 169–183. [CrossRef]
14. Farhi, E.; Goldstone, J.; Gutmann, S. A quantum approximate optimization algorithm. *arXiv* **2014**, arXiv:1411.4028.
15. Lucas, A. Ising formulations of many NP problems. *Front. Phys.* **2014**, *2*, 5. [CrossRef]
16. Nüßlein, J.; Gabor, T.; Linnhoff-Popien, C.; Feld, S. Algorithmic QUBO formulations for k-SAT and Hamiltonian cycles. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Boston, MA, USA, 9–13 July 2022; pp. 2240–2246.
17. Gabor, T.; Zielinski, S.; Feld, S.; Roch, C.; Seidel, C.; Neukart, F.; Galter, I.; Mauerer, W.; Linnhoff-Popien, C. Assessing solution quality of 3SAT on a quantum annealing platform. In Proceedings of the Quantum Technology and Optimization Problems: First International Workshop, QTOP 2019, Munich, Germany, 18 March 2019; pp. 23–35.
18. Sax, I.; Feld, S.; Zielinski, S.; Gabor, T.; Linnhoff-Popien, C.; Mauerer, W. Approximate approximation on a quantum annealer. In Proceedings of the 17th ACM International Conference on Computing Frontiers, Sicily, Italy, 11–13 May 2020; pp. 108–117.
19. Mandl, A.; Barzen, J.; Bechtold, M.; Leymann, F.; Wild, K. Amplitude amplification-inspired QAOA: Improving the success probability for solving 3SAT. *arXiv* **2023**, arXiv:2303.01183.
20. Ayanzadeh, R.; Halem, M.; Finin, T. Reinforcement quantum annealing: A hybrid quantum learning automata. *Sci. Rep.* **2020**, *10*, 7952. [CrossRef] [PubMed]
21. Choi, V. Adiabatic quantum algorithms for the NP-complete maximum-weight independent set, exact cover and 3SAT problems. *arXiv* **2010**, arXiv:1004.2226.
22. Chancellor, N.; Zohren, S.; Warburton, P.A.; Benjamin, S.C.; Roberts, S. A direct mapping of Max k-SAT and high order parity checks to a chimera graph. *Sci. Rep.* **2016**, *6*, 37107. [CrossRef] [PubMed]
23. Zielinski, S.; Nüßlein, J.; Stein, J.; Gabor, T.; Linnhoff-Popien, C.; Feld, S. Influence of Different 3SAT-to-QUBO Transformations on the Solution Quality of Quantum Annealing: A Benchmark Study. *arXiv* **2023**, arXiv:2305.00720.
24. Moraglio, A.; Georgescu, S.; Sadowski, P. AutoQubo: Data-driven automatic QUBO generation. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Boston, MA, USA, 9–13 July 2022; pp. 2232–2239.
25. Pauckert, J.; Ayodele, M.; García, M.D.; Georgescu, S.; Parizy, M. AutoQUBO v2: Towards Efficient and Effective QUBO Formulations for Ising Machines. In Proceedings of the Companion Conference on Genetic and Evolutionary Computation, Lisbon, Portugal, 15–19 July 2023; pp. 227–230.
26. Richoux, F.; Baffier, J.F.; Codognet, P. Learning QUBO Models for Quantum Annealing: A Constraint-based Approach. In Proceedings of the International Conference on Computational Science, Prague, Czech Republic, 3–5 July 2023.
27. Zaman, M.; Tanahashi, K.; Tanaka, S. PyQUBO: Python library for mapping combinatorial optimization problems to QUBO form. *IEEE Trans. Comput.* **2021**, *71*, 838–850. [CrossRef]
28. Cheeseman, P.C.; Kanefsky, B.; Taylor, W.M. Where the really hard problems are. In Proceedings of the IJCAI, Sydney, Australia, 24–30 August 1991; Volume 91, pp. 331–337.
29. Gent, I.P.; Walsh, T. The SAT phase transition. In Proceedings of the ECAI, Amsterdam, The Netherlands, 8–12 August 1994; Volume 94, pp. 105–109.
30. Glover, F.; Kochenberger, G.; Du, Y. A tutorial on formulating and using QUBO models. *arXiv* **2018**, arXiv:1811.11538.

31. Tanahashi, K.; Takayanagi, S.; Motohashi, T.; Tanaka, S. Application of Ising machines and a software development for Ising machines. *J. Phys. Soc. Jpn.* **2019**, *88*, 061010. [CrossRef]
32. Lamm, S.; Schulz, C.; Strash, D.; Williger, R.; Zhang, H. Exactly solving the maximum weight independent set problem on large real-world graphs. In Proceedings of the 2019 Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX), San Diego, CA, USA, 7–8 January 2019; pp. 144–158.
33. Zielinski, S. Pattern QUBO 3SAT Framework. 2023. Available online: https://github.com/ZielinskiSebastian/SAT-QUBO-Framework (accessed on 14 August 2023).
34. Galindo, O.; Kreinovich, V. What is the optimal annealing schedule in quantum annealing. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, Australia, 1–4 December 2020; pp. 963–967.
35. Chen, Y.Q.; Chen, Y.; Lee, C.K.; Zhang, S.; Hsieh, C.Y. Optimizing quantum annealing schedules with Monte Carlo tree search enhanced with neural networks. *Nat. Mach. Intell.* **2022**, *4*, 269–278. [CrossRef]