

Solving (Max) 3-SAT via Quadratic Unconstrained Binary Optimization

Nüßlein, Jonas; Zielinski, Sebastian; Gabor, Thomas; Linnhoff-Popien, Claudia; Feld, Sebastian

DOI

[10.1007/978-3-031-36030-5_3](https://doi.org/10.1007/978-3-031-36030-5_3)

Publication date

2023

Document Version

Final published version

Published in

Computational Science – ICCS 2023 - 23rd International Conference, Proceedings

Citation (APA)

Nüßlein, J., Zielinski, S., Gabor, T., Linnhoff-Popien, C., & Feld, S. (2023). Solving (Max) 3-SAT via Quadratic Unconstrained Binary Optimization. In J. Mikyška, C. de Mulatier, V. V. Krzhizhanovskaya, P. M. A. Soot, M. Paszynski, & J. J. Dongarra (Eds.), *Computational Science – ICCS 2023 - 23rd International Conference, Proceedings* (pp. 34-47). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 14077 LNCS). Springer. https://doi.org/10.1007/978-3-031-36030-5_3

Important note

To cite this publication, please use the final published version (if applicable). Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights. We will remove access to the work immediately and investigate your claim.

Green Open Access added to TU Delft Institutional Repository


'You share, we take care!' - Taverne project

<https://www.openaccess.nl/en/you-share-we-take-care>

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.



Solving (Max) 3-SAT via Quadratic Unconstrained Binary Optimization

Jonas Nüßlein¹ , Sebastian Zielinski¹, Thomas Gabor¹,
Claudia Linnhoff-Popien¹, and Sebastian Feld²

¹ Institute for Informatics, LMU Munich, Munich, Germany
jonas.nuesslein@ifi.lmu.de

² Faculty of Electrical Engineering, Mathematics and Computer Science, TU Delft,
Delft, The Netherlands

Abstract. We introduce a novel approach to translate arbitrary 3-SAT instances to Quadratic Unconstrained Binary Optimization (QUBO) as they are used by quantum annealing (QA) or the quantum approximate optimization algorithm (QAOA). Our approach requires fewer couplings and fewer physical qubits than the current state-of-the-art, which results in higher solution quality. We verified the practical applicability of the approach by testing it on a D-Wave quantum annealer.

Keywords: QUBO · quantum annealing · satisfiability · 3-SAT

1 Introduction

In recent years, many well-known optimization and decision problems have been translated to the model of quadratic unconstrained binary optimization (QUBO) [13, 18]. The main motivation behind this is that QUBO models can be used as a problem specification for various early quantum algorithms, most notably the quantum approximate optimization algorithm (QAOA) [9, 24] and quantum annealing (QA) [16, 17]. Current quantum computers are noisy and limited in size; thus it is important to encode problems as efficiently as possible. However, quantum hardware is conjectured to further grow in capability and a first demo application recently suggested that it might already have a substantial advantage over classical hardware for specific tasks [1].

The most promising problems to be solved using quantum algorithms certainly include problems of the complexity class NP-hard, which are hard to solve for classical computers (unless $P = NP$) [7, 11]. Many NP-hard problems like scheduling [23], quadratic assignment [19], or travelling salesman [10] are of immense practical importance and practical instances often challenge current computing hardware. Thus, the eventual benefit of making these kinds of problems faster to solve may be especially appealing.

The canonical problem for the class NP-complete is 3-satisfiability (3-SAT), which we focus on in this paper [8]. A 3-SAT instance is a formula in Boolean algebra and its solution is the binary answer to whether the formula is satisfiable.

Our contributions in this paper are:

- We present two novel 3-SAT-to-QUBO translations: NÜSSLEIN^{2n+m} and NÜSSLEIN^{n+m}
- We empirically show that NÜSSLEIN^{2n+m} performs slightly better than CHANCELLOR^{n+m} despite the bigger QUBO matrix
- We show that NÜSSLEIN^{n+m} requires fewer couplings and fewer physical qubits than the current state-of-the-art approach CHANCELLOR^{n+m}
- We empirically show that NÜSSLEIN^{n+m} performs best, compared to three other 3-SAT-to-QUBO translations

2 Foundations

In this section, we introduce the mathematical foundations of the problems involved in the translation algorithms: 3-SAT and QUBO.

2.1 Satisfiability Problems

The satisfiability problem (SAT) of propositional logic is informally defined as follows: Given a Boolean formula, is there any assignment of the involved variables so that the formula is reduced to “true”? The problem occurs in every application involving complex constraints or reasoning, like (software) product lines, the tracing of software dependencies, or formal methods [12].

All SAT problem instances can be reduced with only polynomial overhead to a specific type of SAT problem called 3-SAT, in which the input propositional logic formula has to be in conjunctive normal form with all of the disjunctions containing exactly three literals.

Definition 1 3-SAT. *A 3-SAT instance with n variables and m clauses is given as (i) a list of variables $(v_j)_{0 \leq j \leq n-1}$, from which a list of literals $(l_i)_{0 \leq i \leq 3m-1}$ can be built of the form*

$$l_i \in \bigcup_{0 \leq j \leq n-1} \{v_j, \neg v_j\},$$

and (ii) a list of clauses $(c_k)_{0 \leq k \leq m-1}$ of the form

$$c_k = (l_{3k} \vee l_{3k+1} \vee l_{3k+2}).$$

A given 3-SAT instance is satisfiable iff there exists a variable assignment given by the structure $(v_j \mapsto b_j)_{0 \leq j \leq n-1}$ with $b_j \in \{\top, \perp\}$ so that

$$\bigwedge_{0 \leq k \leq m-1} c_k$$

reduces to \top when interpreting all logical operators as is common. The problem of deciding whether a given 3-SAT instance is satisfiable is called 3-SAT.

For example, we may write a 3-SAT instance as Boolean formula $\mathcal{F} = (a \vee b \vee c) \wedge (a \vee \neg c \vee \neg d)$ consisting of $m = 2$ clauses and featuring the $n = 4$ distinct variables $\{a, b, c, d\}$. Obviously, \mathcal{F} is satisfiable, for example via the variable assignment $(a \mapsto \perp, b \mapsto \top, c \mapsto \top, d \mapsto \perp)$.

3-SAT was the first problem to be shown to be NP-complete, which means that all problems in NP can be reduced to 3-SAT [8]. In fact, as many proofs for NP-completeness for other problems build upon their reduction to 3-SAT, 3-SAT solvers can be used as tools to solve many different decision problems.

As 3-SAT is central to many proofs of NP-completeness, it is somewhat surprising that when we generate random 3-SAT instances with random amounts of variables n and clauses m , most of these instances will be really easy to solve for standard SAT solvers. It is only as the ratio of clauses per variable approaches $\frac{m}{n} \approx 4.2$ that we can see the problems take exponential computing time. Knowing that many 3-SAT instances are relatively easy to solve even for classical computers, we focus our attention regarding new methods (like quantum-based ones) on the critical 3-SAT instances with $\frac{m}{n} \approx 4.2$.

MAX-3-SAT is an optimization problem that corresponds to the decision problem 3-SAT. Instead of checking whether an assignment exists that fulfils the whole formula, i.e., reduces all clauses individually to \top , we try to find the assignment that fulfils as many clauses as possible. Note that MAX-3-SAT is a generalization of 3-SAT as MAX-3-SAT's optimal result is an assignment that fulfils all clauses and thus proves the satisfiability of the whole formula.

Definition 2 MAX-3-SAT. A MAX-3-SAT instance is given the same way as a 3-SAT instance (cf. Definition 1). The objective of a MAX-3-SAT instance is to find a variable assignment of the structure $(v_j \mapsto b_j)_{0 \leq j \leq n-1}$ with $b_j \in \{\top, \perp\}$ so to

$$\text{maximize} \quad \sum_{k=0}^{m-1} \begin{cases} 1 & \text{if } c_k \text{ reduces to } \top, \\ 0 & \text{otherwise.} \end{cases}$$

2.2 Quadratic Unconstrained Binary Optimization

In quadratic unconstrained binary optimization (QUBO) we are looking for a binary vector $\mathbf{x} = \langle x_i \rangle_{0 \leq i \leq k-1}$ of length k that minimizes the value of a formula that at most contains quadratic terms in x .

Definition 3 QUBO. A QUBO instance with k variables is given as a $k \times k$ matrix $Q \in \mathbb{R}^{k \times k}$. The objective of a QUBO instance is to find a binary vector $\mathbf{x} \in \mathbb{B}^k$ so to

$$\text{minimize} \quad H(\mathbf{x}) = \sum_i Q_{ii}x_i + \sum_{i < j} Q_{ij}x_ix_j.$$

$H(\mathbf{x})$ is also called the *energy* of a QUBO solution \mathbf{x} . Note that the lower triangle of the matrix Q is always empty (since its values do not occur in the formula for the energy H). Finding the ideal solution vector \mathbf{x} of a given QUBO Q is NP-hard.

When solving QUBO instances using a quantum annealer, the solution vector \mathbf{x} is mapped to a set of qubits. These qubits have connections whose strength can be manipulated to emulate the values in the QUBO matrix. As the limiting factor in current hardware is the size of problems that can be solved, we seek translations to QUBO that require as few qubits (i.e., minimal size of the QUBO matrix) and as few connections between them (i.e., minimal density within the QUBO matrix) as possible.

3 Related Work

There are currently two main approaches for translating 3-SAT to QUBO, which we refer to as CHANCELLOR ^{$n+m$} [5] and CHOI ^{$3m$} [6]. We will review them in more detail in Subsects. 3.1 and 3.2 respectively.

In [12], the authors examined the critical region of the problem domain for 3-SAT, i.e., instances with $\frac{m}{n} \approx 4.2$. They observed that the clause-to-variable ratio has a great impact on the solution quality even on the quantum annealers.

Quantum annealing has previously been regarded as a solution to satisfiability problems: [2] focuses on embedding an originally SAT-related QUBO into the architecture of the most common quantum annealing chip. [20] shows a method to derive formulation for the optimization energy and proves mathematical bounds for the mapping of general k -SAT problems. Similarly, [14] shows an approach justifying feasibility but provides no empirical data. In [15], Grover’s search algorithm was used to solve k -SAT. In [21], a QUBO formulation for k -SAT is proposed which only scales logarithmically in k compared to the linear scaling in [6] and [5]. In [22], a method is proposed to not hard-code a QUBO to SAT translation but to learn it using gradient-based methods.

3.1 Chancellor ^{$n+m$}

Let $a_i^{(l)}$ be the i -th literal of clause $a^{(l)}$. The idea in [5] is to present a QUBO formulation for an arbitrary clause that assigns the energy g to the *one* variable assignment which does not fulfil the clause and the energy 0 to all other possible variable assignments. The energy spectrum is therefore given by:

$$Spec(\{a^{(l)}\}) = \begin{cases} g & a_i^{(l)} = 0 \quad \forall i, \\ 0 & otherwise. \end{cases}$$

Thus we can create the QUBO formulation for the whole 3-SAT formula by superimposing all clause-formulations: $H = \sum_l Spec(\{a^{(l)}\})$. For $g > 0$ the minimum energy bit-string will always be the one which satisfies the most clauses.

To move from logical values to spin variables, one can map each logical variable $a_i = 0$ to a spin variable with value $\sigma_i^z = -1$ and each logical variable $a_i = 1$ to a spin variable with value $\sigma_i^z = +1$. Negation of the logical variable is then implemented through gauges on the spin variables. More precisely, a_i is mapped to $c(i)\sigma_i^z$ with $c(i) = 1$ and $\neg a_i$ to $c(i)\sigma_i^z$ with $c(i) = -1$.

The authors subsequently present the clause-formulation in the following way: The energy spectrum of the clause $(a_1 \vee a_2 \vee a_3)$ can be rewritten as $a_1 + a_2 + a_3 - a_1a_2 - a_1a_3 - a_2a_3 + a_1a_2a_3$ (with $a_i \in \{-1, +1\}$). The terms $a_1, a_2, a_3, a_1a_2, a_1a_3$ and a_2a_3 can be directly inserted into the Ising Hamiltonian. For the triple term $a_1a_2a_3$, however, an ancilla qubit is necessary. The authors then present an Ising Hamiltonian for the triple term:

$$H = h \sum_{i=1}^3 c(i)\sigma_i^z + J^a \sum_{i=1}^3 c(i)\sigma_i^z\sigma_a^z + h^a\sigma_a^z$$

in which up to the gauge choice $c(i) \in \{-1, 1\}$ the 3 variables σ_i^z are coupled with equal strength J^a to the same ancilla spin variable σ_a^z .

There are some constraints for the choice of the hyperparameters h , J^a , h^a , and J . We chose $h = g = 1, h^a = 2h = 2, J = 5$ and thus $J^a = 2J = 10$ as values for the variables in the ‘‘special cases’’ section of Chancellor et al. [5]. It is important to note that the choice of these values has no influence on the number of couplings needed. Any clause-translation will produce a fully-connected Ising/QUBO matrix (Note that Ising and QUBO are isomorphic).

For each clause exactly one ancilla qubit C_i is needed. Thus the whole QUBO matrix will have size $n + m$. The 3-SAT formula $(\neg a \vee \neg b \vee \neg c) \wedge (a \vee b \vee c)$ would, for example, be represented by the QUBO matrix in Table 1.

Table 1. QUBO matrix using CHANCELLOR ^{$n+m$} for the 3-SAT formula $(\neg a \vee \neg b \vee \neg c) \wedge (a \vee b \vee c)$.

	a	b	c	C_1	C_2
a	-88	48	48	40	40
b		-88	48	40	40
c			-88	40	40
C_1				-56	0
C_2					-64

3.2 Choi^{3m}

Choi [6] provides a translation of 3-SAT to QUBO that takes up $3m$ qubits, i.e., three qubits per clause in the original 3-SAT formula (or one qubit per literal). It is inspired by the maximum independent set problem (to which 3-SAT is first reduced, then to QUBO). Given a 3-SAT instance with m clauses and n variables, CHOI^{3m} reserves a qubit $x_{k,i}, 0 \leq k < m, 0 \leq i \leq 2$, for every literal. Thus CHOI^{3m} needs $3m$ qubits in total. One can interpret a solution candidate x for this QUBO formulation in the following way:

- If $x_{k,i} = 1$ and the corresponding literal $l_{3k+i} = v$ for some variable v , then we add the assignment $(v \mapsto \top)$ to the solution candidate for 3-SAT.
- If $x_{k,i} = 1$ and the corresponding literal $l_{3k+i} = \neg v$ for some variable v , then we add the assignment $(v \mapsto \perp)$ to the solution candidate for 3-SAT.
- If $x_{k,i} = 0$, then we do nothing.

Note that a solution candidate for the QUBO may thus be illegitimate from the 3-SAT perspective when it assigns different truth values to the same variable. Further note that CHOI^{3m} , even when returning the perfectly optimal solution, does not necessarily assign a truth value to every variable that occurs in the original formula.

For the detailed algorithm, we refer to [6] and will instead provide a small example. Note that the incentive and penalty values X, Y, Z can be chosen rather freely as long as $Y > 2|X|$ and $Z > 2|X|$. Given the example 3-SAT instance $(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$, we can then write a QUBO matrix as follows:

$$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$$

Q	$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{1,0}$	$x_{1,1}$	$x_{1,2}$
$x_{0,0}$	$-X$	Y	Y			
$x_{0,1}$		$-X$	Y			
$x_{0,2}$			$-X$			Z
$x_{1,0}$				$-X$	Y	Y
$x_{1,1}$					$-X$	Y
$x_{1,2}$						$-X$

Intuitively, we need to penalize setting a pair of qubits from the same clause (Y) and penalize setting a pair of qubits which correspond to contradicting literals of the same variable (Z). Since so far we only assigned penalties, we need to set negative energy values on the diagonal ($-X$) in order to incentivize setting any qubits at all (and avoid the trivial solution $\mathbf{x} = \mathbf{0}$).

4 Approaches

We now describe two new approaches for translating a given 3-SAT instance to QUBO. We introduce a new approach NÜSSLEIN^{2n+m} in Sect. 4.1, which uses $2n+m$ logical qubits, where n is the number of variables and m is the number of clauses. In Sect. 4.2 we then propose another formulation NÜSSLEIN^{n+m} , which requires $n+m$ qubits. This is on par with the state-of-the-art CHANCELLOR^{n+m} ; however, NÜSSLEIN^{n+m} uses fewer couplings, which leads to a reduction of physical qubits.

4.1 A $2n + m$ Approach

We now introduce a novel approach for the translation of 3-SAT to QUBO: NÜSSLEIN^{2n+m}. Like CHOI^{3m} and CHANCELLOR^{n+m}, NÜSSLEIN^{2n+m} actually solves MAX-3-SAT by trying to accumulate as many solvable clauses as possible. We build on the idea of [21] to use an algorithm to describe the QUBO translation instead of an arithmetic notation.

We use the qubits in the following way:

- For each variable v_j , $0 \leq j \leq n - 1$, occurring in the 3-SAT instance, we use two qubits to encode if the variable is to be assigned \top or if the variable is to be assigned \perp . Thus, $(v_j \mapsto \top)$ occurs in the variable assignment if $x_{2j} = 1$. Likewise, $(v_j \mapsto \perp)$ occurs in the variable assignment if $x_{2j+1} = 1$. Note that assigning both $v_{2j} = v_{2j+1}$ the same value makes for an illegitimate 3-SAT solution candidate.
- Beyond those qubits, we further use one qubit for every clause in the 3-SAT instance.

Effectively, the approach then uses $2n + m$ qubits for a 3-SAT instance with n variables and m clauses. This may be less or more than the $3m$ qubits used in CHOI^{3m}; however, consider that difficult 3-SAT instances are categorized by $\frac{m}{n} \approx 4.2$ (cf. Sect. 2). Thus, for the 3-SAT instances which actually require extensive computations on classical computers, NÜSSLEIN^{2n+m} manages to generate substantially smaller matrices. For the detailed instructions of Algorithm 1, we first need to introduce the following definitions:

- We write $L = (v_0, \neg v_0, \dots, v_{n-1}, \neg v_{n-1})$ for the list containing all possible literals given variables $(v_j)_{0 \leq j \leq n-1}$. Note that $|L| = 2n$.
- We write $v_j \in c_k$ when clause c_k contains a literal of the form v_j . Likewise, we write $\neg v_j \in c_k$ when c_k contains a literal of the form $\neg v_j$. We subsequently write $L_i \in c_k$ when c_k contains the literal L_i .
- We define

$$R(L_i) = \sum_{k=0}^{m-1} \begin{cases} 1 & \text{if } L_i \in c_k, \\ 0 & \text{otherwise.} \end{cases}$$

Thus $R(L_i)$ is counting how often the literal L_i occurs in the formula.

- We define

$$R(L_i, L_{i'}) = \sum_{k=0}^{m-1} \begin{cases} 1 & \text{if } L_i \in c_k \text{ and } L_{i'} \in c_k, \\ 0 & \text{otherwise.} \end{cases}$$

Thus $R(L_i, L_{i'})$ is the number of occurrences of the literals L_i and $L_{i'}$ together in the same clause.

Intuitively, NÜSSLEIN^{2n+m} (cf. Algorithm 1) encodes how many clauses are fulfilled by the solution. For example, if the minimal energy H^* of a given NÜSSLEIN^{2n+m}-QUBO is -20 this means that 20 clauses are fulfilled. If the formula, however, has more than 20 clauses this means that the formula is not

Algorithm 1. NÜSSLEIN^{2n+m}

```

1: procedure NÜSSLEIN2n+m
2:    $Q = \mathbf{0} \in \mathbb{R}^{2n+m \times 2n+m}$ 
3:   for  $i := 0$  to  $2n + m$  do
4:     for  $j := i$  to  $2n + m$  do
5:       if  $i = j$  and  $j < 2n$  then
6:          $Q_{ij} := -R(L_i)$ 
7:       else if  $i = j$  and  $j \geq 2n$  then
8:          $Q_{ij} := 2$ 
9:       else if  $j < 2n$  and  $j - i = 1$  and  $i \bmod 2 = 0$  then
10:         $Q_{ij} := m + 1$ 
11:       else if  $i < 2n$  and  $j < 2n$  then
12:         $Q_{ij} := R(L_i, L_j)$ 
13:       else if  $j \geq 2n$  and  $i < 2n$  and  $l_i$  in  $c_{j-2n}$  then
14:         $Q_{ij} = -1$ 
15:       end if
16:     end for
17:   end for
18:   return  $Q$ 
19: end procedure

```

satisfiable. We can consider the example formula $(a \vee b \vee \neg c) \wedge (a \vee \neg b \vee \neg c)$ and its translation to QUBO using NÜSSLEIN^{2n+m}:

$$(a \vee b \vee \neg c) \wedge (a \vee \neg b \vee \neg c)$$

Q	a	$\neg a$	b	$\neg b$	c	$\neg c$	$(a \vee b \vee \neg c)$	$(a \vee \neg b \vee \neg c)$
a	-2	3	1	1	0	2	-1	-1
$\neg a$		0	0	0	0	0	0	0
b			-1	3	0	1	-1	0
$\neg b$				-1	0	1	0	-1
c					0	3	0	0
$\neg c$						-2	-1	-1
$(a \vee b \vee \neg c)$							2	0
$(a \vee \neg b \vee \neg c)$								2

4.2 An $n + m$ Approach

In this section we present NÜSSLEIN^{n+m}, which is a 3-SAT (again actually MAX-3-SAT) to QUBO translation, which only requires $n + m$ logical qubits. This is on

par with CHANCELLOR^{n+m}. However, we will show that our approach requires fewer couplings, which leads to a reduction of needed physical qubits in the hardware embedding.

We use the qubits in the following way:

- For each variable $v_j, 0 \leq j \leq n - 1$, occurring in the 3-SAT instance, we use one qubit to encode the value it is assigned. Thus, $(v_j \mapsto \top)$ occurs in the variable assignment iff $x_j = 1$. This implies that $(v_j \mapsto \perp)$ occurs in the variable assignment iff $x_j = 0$.
- Beyond those qubits, we again use one qubit for every clause in the 3-SAT instance.

For the algorithm, we start with an empty QUBO matrix as a canvas and then add specific patterns of values for each clause. As these pattern stack, we acquire the final value of Q_{ij} as a sum of all stacked values. The algorithm thus needs to iterate over all clauses and repeatedly update the QUBO matrix while doing so. As we need to look at each clause individually, we can assume without loss of generality that all clauses are sorted, i.e., all negated literals appear as far towards the back of the clause as possible. This leaves us with only four possible patterns for clauses:

$$(a \vee b \vee c), (a \vee b \vee \neg c), (a \vee \neg b \neg c), (\neg a \vee \neg b \vee \neg c)$$

We now want to arrange the energy levels for each of the four cases such that a satisfied clause (no matter in which way it was satisfied, i.e., with one literal, with two, or with three) has the energy H^* and the *one* state which does not satisfy the clause has the energy $H^+ = H^* + 1$. See Table 2 for all pattern matrices that might occur. The final QUBO matrix is then constructed by adding the pattern matrices' values to the cells in the QUBO matrix that correspond to the involved variables. For a 3-SAT formula with p clauses where there are no negated literals and q clauses where there are only negated literals, a variable assignment that satisfies the entire formula has the energy $H^* = -p - q$.

We can now consider the example formula $(a \vee b \vee c) \wedge (a \vee \neg b \vee \neg c)$ and its translation to QUBO using NÜSSLEIN^{n+m}:

$$(a \vee b \vee c) \wedge (a \vee \neg b \vee \neg c)$$

Q	a	b	c	$(a \vee b)$	$(a \vee \neg b)$
a	$0 + 2$	$2 - 2$	$0 + 0$	-2	-2
b		$0 + 0$	$0 + 0$	-2	2
c			$-1 + 1$	1	-1
$(a \vee b)$				1	0
$(a \vee \neg b)$					0

Table 2. Pattern matrices for the four different types of clauses.

$(a \vee b \vee c), H^* = -1$	$(b) (a \vee b \vee \neg c), H^* = 0$																																																																	
<table style="width: 100%; border-collapse: collapse;"> <tr><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">$(a \vee b)$</td></tr> <tr><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">a</td><td style="border: 1px solid black;">b</td><td style="border: 1px solid black;">c</td><td style="border: 1px solid black;"></td></tr> <tr><td style="border: 1px solid black;">a</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">2</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">-2</td></tr> <tr><td style="border: 1px solid black;">b</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">-2</td></tr> <tr><td style="border: 1px solid black;">c</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">-1</td><td style="border: 1px solid black;">1</td></tr> <tr><td style="border: 1px solid black;">$(a \vee b)$</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">1</td></tr> </table>					$(a \vee b)$		a	b	c		a		2		-2	b				-2	c			-1	1	$(a \vee b)$				1	<table style="width: 100%; border-collapse: collapse;"> <tr><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">$(a \vee b)$</td></tr> <tr><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">a</td><td style="border: 1px solid black;">b</td><td style="border: 1px solid black;">c</td><td style="border: 1px solid black;"></td></tr> <tr><td style="border: 1px solid black;">a</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">2</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">-2</td></tr> <tr><td style="border: 1px solid black;">b</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">-2</td></tr> <tr><td style="border: 1px solid black;">c</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">1</td><td style="border: 1px solid black;">-1</td></tr> <tr><td style="border: 1px solid black;">$(a \vee b)$</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">2</td></tr> </table>					$(a \vee b)$		a	b	c		a		2		-2	b				-2	c			1	-1	$(a \vee b)$				2					
				$(a \vee b)$																																																														
	a	b	c																																																															
a		2		-2																																																														
b				-2																																																														
c			-1	1																																																														
$(a \vee b)$				1																																																														
				$(a \vee b)$																																																														
	a	b	c																																																															
a		2		-2																																																														
b				-2																																																														
c			1	-1																																																														
$(a \vee b)$				2																																																														
$(c) (a \vee \neg b \vee \neg c), H^* = 0$	$(d) (\neg a \vee \neg b \vee \neg c), H^* = 1$																																																																	
<table style="width: 100%; border-collapse: collapse;"> <tr><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">$(a \vee \neg b)$</td></tr> <tr><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">a</td><td style="border: 1px solid black;">b</td><td style="border: 1px solid black;">c</td><td style="border: 1px solid black;"></td></tr> <tr><td style="border: 1px solid black;">a</td><td style="border: 1px solid black;">2</td><td style="border: 1px solid black;">-2</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">-2</td></tr> <tr><td style="border: 1px solid black;">b</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">2</td></tr> <tr><td style="border: 1px solid black;">c</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">1</td><td style="border: 1px solid black;">-1</td></tr> <tr><td style="border: 1px solid black;">$(a \vee \neg b)$</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td></tr> </table>					$(a \vee \neg b)$		a	b	c		a	2	-2		-2	b				2	c			1	-1	$(a \vee \neg b)$					<table style="width: 100%; border-collapse: collapse;"> <tr><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">$(\neg a \wedge \neg b \wedge \neg c)$</td></tr> <tr><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">a</td><td style="border: 1px solid black;">b</td><td style="border: 1px solid black;">c</td><td style="border: 1px solid black;"></td></tr> <tr><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td></tr> <tr><td style="border: 1px solid black;">a</td><td style="border: 1px solid black;">-1</td><td style="border: 1px solid black;">1</td><td style="border: 1px solid black;">1</td><td style="border: 1px solid black;">1</td></tr> <tr><td style="border: 1px solid black;">b</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">-1</td><td style="border: 1px solid black;">1</td><td style="border: 1px solid black;">1</td></tr> <tr><td style="border: 1px solid black;">c</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">-1</td><td style="border: 1px solid black;">1</td></tr> <tr><td style="border: 1px solid black;">$(\neg a \wedge \neg b \wedge \neg c)$</td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;"></td><td style="border: 1px solid black;">-1</td></tr> </table>					$(\neg a \wedge \neg b \wedge \neg c)$		a	b	c							a	-1	1	1	1	b		-1	1	1	c			-1	1	$(\neg a \wedge \neg b \wedge \neg c)$				-1
				$(a \vee \neg b)$																																																														
	a	b	c																																																															
a	2	-2		-2																																																														
b				2																																																														
c			1	-1																																																														
$(a \vee \neg b)$																																																																		
				$(\neg a \wedge \neg b \wedge \neg c)$																																																														
	a	b	c																																																															
a	-1	1	1	1																																																														
b		-1	1	1																																																														
c			-1	1																																																														
$(\neg a \wedge \neg b \wedge \neg c)$				-1																																																														

A possible optimal solution to this QUBO would be $\mathbf{x} = \langle 1, 0, 0, 1, 1 \rangle$, which corresponds to the variable assignment: ($a \mapsto \top, b \mapsto \perp, c \mapsto \perp$) with the energy $H(\mathbf{x}) = -1$. Note that this QUBO matrix uses 5 logical qubits and 6 couplings (non-zero weights in the QUBO matrix). We can compare that to the CHANCELLOR ^{$n+m$} formulation, which requires 5 logical qubits as well but 9 couplings:

$$(a \vee b \vee c) \wedge (a \vee \neg b \vee \neg c)$$

Q	a	b	c	C_1	C_2
a	-88	40	40	40	40
b		-88	48	40	40
c			-88	40	40
C_1				-64	0
C_2					-64

Another notable feature of NÜSSLEIN ^{$n+m$} is the possibility to use the same clause-qubit for more than one clause. For example in the formula $(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$ the logical sub-formula $(a \vee b)$ appears in both clauses thus we just need one clause-qubit instead of two. In total, we thus would need 4 logical qubits instead of 5.

5 Empirical Evaluation

To empirically verify that NÜSSEIN^{n+m} requires fewer couplings than CHANCELLOR^{n+m} we created random 3-SAT formulas, applied both approaches, and counted the number of non-zero elements in the corresponding QUBO matrices. The results are shown in Fig. 1. The x-axis describes the number of variables V in the 3-SAT formula. We then created random formulas with $\lceil 4.2V \rceil$ clauses. As can be seen in the charts, for both approaches the number of non-zero couplings in the QUBO matrices scales linearly in the number of variables V of the 3-SAT formula. However, NÜSSEIN^{n+m} only requires roughly 0.7 of the couplings that CHANCELLOR^{n+m} needs.

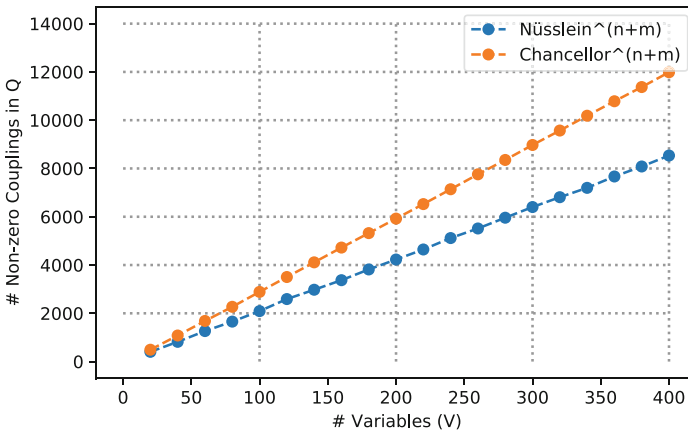


Fig. 1. Relation of the number of variables in the 3-SAT formula to the number of non-zero couplings in the QUBO matrix for the approaches CHANCELLOR^{n+m} and NÜSSEIN^{n+m} .

In the next experiment, we evaluated how this reduction of couplings translates to a reduction of physical qubits. Note that both approaches NÜSSEIN^{n+m} and CHANCELLOR^{n+m} require $n+m$ *logical* qubits. However, to run a QUBO on a quantum annealer the QUBO has to be embedded into the hardware graph, which currently follows the Pegasus graph design [3]. We again created random 3-SAT formulas for different V , applied both approaches to create the corresponding QUBO matrices and then ran the minorminer to find an embedding [4]. Finally, we counted how many *physical* qubits were needed. The results (Fig. 2) show that for both approaches the number of physical qubits scales linearly with V but the chart of CHANCELLOR^{n+m} has again a bigger gradient than the chart of NÜSSEIN^{n+m} . The line represents the median of 20 formulas and the shaded areas enclose the 0.25 and 0.75 quantiles.

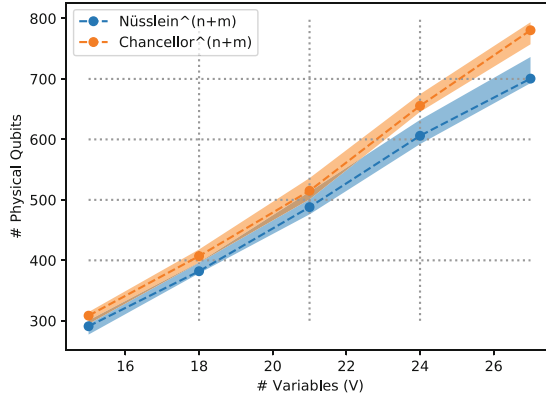


Fig. 2. Relation of the number of variables in the 3-SAT formula to the number of needed physical qubits for the approaches CHANCELLOR^{n+m} and NÜSSLEIN^{n+m} . The shaded areas enclose the 0.25 and 0.75 quantiles.

In a final experiment, we created random 3-SAT formulas and solved them with all four methods on the D-Wave Quantum Annealer. We tested three sizes for the 3-SAT formula and for each we created 20 random formulas. Table 3 shows the mean number of fulfilled clauses with the best-found variable assignment. For example for the size $(V = 5, C = 21)$ we created a random formula and solved it using NÜSSLEIN^{2n+m} on the D-Wave. For the best answer of the D-Wave, we calculated the variable assignment and how many clauses are fulfilled with this assignment. We repeated this procedure for 20 3-SAT formulas. As can be seen, NÜSSLEIN^{n+m} was the best approach for every size of the formula. Another very interesting result is that CHOI^{3m} was mostly better than NÜSSLEIN^{2n+m} and NÜSSLEIN^{2n+m} was mostly better than CHANCELLOR^{n+m} which indicates that the size of the QUBO matrix is not an optimal predictor for performance. The code for all four approaches can be found here: <https://github.com/JonasNuesslein/3SAT-with-QUBO>.

Table 3. Performance of four 3-SAT to QUBO translations on random formulas. The values represent the mean number of fulfilled clauses of the best-found solution vector.

	$(V = 5, C = 21)$	$(V = 10, C = 42)$	$(V = 12, C = 50)$
NÜSSLEIN^{2n+m}	20.4	39.0	45.0
NÜSSLEIN^{n+m}	20.6	41.2	49.0
CHANCELLOR^{n+m}	18.6	37.8	47.0
CHOI^{3m}	20.0	39.8	47.2

6 Conclusion and Future Work

In this paper, we presented two new approaches to translate 3-SAT instances to QUBO. Despite the smaller size of the QUBO, the first approach NÜSSLEIN^{2n+m} showed worse results than CHOI^{3m} in the experiments, which indicates that the size of a QUBO is not an optimal predictor for performance. For the other approach NÜSSLEIN^{n+m}, we showed that it requires fewer couplings and fewer physical qubits than the current state-of-the-art CHANCELLOR^{n+m}. We empirically verified that NÜSSLEIN^{n+m} performs best compared to three other 3-SAT to QUBO translations. The structure of the NÜSSLEIN^{n+m} approach also shows a new paradigm in constructing QUBO translations: We did not derive a formulation from the original problem by adapting the mathematical framework; the QUBO matrix of NÜSSLEIN^{n+m} was instead constructed from the ground up with the sole goal of mirroring 3-SAT's global optimum. We hope that NÜSSLEIN^{n+m} can thus also inspire more new QUBO translations in the future.

Regarding 3-SAT, it needs to be further investigated whether in general or for special cases even more favorable QUBO formulations for 3-SAT exist. This investigation could also be formulated as an optimization problem (more precisely as Integer Linear Program), where all solutions of the 3-SAT together with the energetically most favorable choice of auxiliary qubits must have the energy H^* and all non-solutions together with the energetically most favourable choice of auxiliary qubits must have an energy $H^+ > H^*$. However, the choice of the “most energetically favorable auxiliary qubits can be formulated by a series of linear inequalities (all other choices of auxiliary qubits with the same variable assignment must have a greater or equal energy). Following this approach, we may even be able to automatically generate new and efficient QUBO translations for practically relevant problems.

References

1. Arute, F., et al.: Quantum supremacy using a programmable superconducting processor. *Nature* **574**(7779), 505–510 (2019)
2. Bian, Z., Chudak, F., Macready, W., Roy, A., Sebastiani, R., Varotti, S.: Solving SAT and MaxSAT with a quantum annealer: foundations and a preliminary report. In: Dixon, C., Finger, M. (eds.) *FroCoS 2017*. LNCS (LNAI), vol. 10483, pp. 153–171. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66167-4_9
3. Boothby, K., Bunyk, P., Raymond, J., Roy, A.: Next-generation topology of D-Wave quantum processors. arXiv preprint [arXiv:2003.00133](https://arxiv.org/abs/2003.00133) (2020)
4. Cai, J., Macready, W.G., Roy, A.: A practical heuristic for finding graph minors. arXiv preprint [arXiv:1406.2741](https://arxiv.org/abs/1406.2741) (2014)
5. 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.* **6**(1), 1–9 (2016)
6. Choi, V.: Adiabatic quantum algorithms for the NP-complete maximum-weight independent set, exact cover and 3SAT problems. arXiv preprint [arXiv:1004.2226](https://arxiv.org/abs/1004.2226) (2010)

7. Cook, S.: The P versus NP problem. In: The Millennium Prize Problems, pp. 87–104 (2006)
8. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing. ACM (1971)
9. Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm. arXiv preprint [arXiv:1411.4028](https://arxiv.org/abs/1411.4028) (2014)
10. Feld, S., et al.: A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer. arXiv preprint [arXiv:1811.07403](https://arxiv.org/abs/1811.07403) (2018)
11. Fortnow, L.: The status of the P versus NP problem. *Commun. ACM* **52**(9), 78–86 (2009)
12. Gabor, T., et al.: Assessing solution quality of 3SAT on a quantum annealing platform. In: Feld, S., Linnhoff-Popien, C. (eds.) QTOP 2019. LNCS, vol. 11413, pp. 23–35. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-14082-3_3
13. Glover, F., Kochenberger, G., Du, Y.: A tutorial on formulating and using QUBO models. arXiv preprint [arXiv:1811.11538](https://arxiv.org/abs/1811.11538) (2018)
14. Hen, I., Spedalieri, F.M.: Quantum annealing for constrained optimization. *Phys. Rev. Appl.* **5**(3), 034007 (2016)
15. Hogg, T.: Adiabatic quantum computing for random satisfiability problems. *Phys. Rev. A* **67**(2), 022314 (2003)
16. Johnson, M.W., et al.: Quantum annealing with manufactured spins. *Nature* **473**(7346), 194–198 (2011)
17. Kadowaki, T., Nishimori, H.: Quantum annealing in the transverse Ising model. *Phys. Rev. E* **58**(5), 5355 (1998)
18. Lucas, A.: Ising formulations of many np problems. *Front. Phys.* **2**, 5 (2014)
19. McGeoch, C.C., Wang, C.: Experimental evaluation of an adiabatic quantum system for combinatorial optimization. In: Proceedings of the ACM International Conference on Computing Frontiers, pp. 1–11 (2013)
20. Mooney, G.J., Tonetto, S.U., Hill, C.D., Hollenberg, L.C.: Mapping NP-hard problems to restricted adiabatic quantum architectures. arXiv preprint [arXiv:1911.00249](https://arxiv.org/abs/1911.00249) (2019)
21. Nüßlein, J., Gabor, T., Linnhoff-Popien, C., Feld, S.: Algorithmic QUBO formulations for K-SAT and Hamiltonian cycles. arXiv preprint [arXiv:2204.13539](https://arxiv.org/abs/2204.13539) (2022)
22. Nüßlein, J., Roch, C., Gabor, T., Linnhoff-Popien, C., Feld, S.: Black box optimization using QUBO and the cross entropy method. arXiv preprint [arXiv:2206.12510](https://arxiv.org/abs/2206.12510) (2022)
23. Venturelli, D., Marchand, D.J., Rojo, G.: Quantum annealing implementation of job-shop scheduling. arXiv preprint [arXiv:1506.08479](https://arxiv.org/abs/1506.08479) (2015)
24. Zahedinejad, E., Zaribafiyani, A.: Combinatorial optimization on gate model quantum computers: a survey. arXiv preprint [arXiv:1708.05294](https://arxiv.org/abs/1708.05294) (2017)