Implementing 3-SAT Gadgets for Quantum Annealers with random instances

Pol Rodríguez Farrés[1,⋆], Rocco Ballester[2,1,⋆], Carlos Ansótegui[3], Jordi Levy[1], and Jesus Cerquides[1]

[1] IIIA-CSIC, Cerdanyola, Spain
`pol.rofa@gmail.com,{cerquide,levy}@iiia.csic.es`
[2] Enzyme Advising Group, Barcelona, Spain
`rocco.ballester@enzyme.biz`
[3] Universitat de Lleida (DIEI, UdL), Lleida, Spain
`carlos@diei.udl.cat`

Abstract. The Maximum Boolean Satisfiability Problem (also known as the Max-SAT problem) is the problem of determining the maximum number of disjunctive clauses that can be satisfied (i.e., made true) by an assignment of truth values to the formula's variables. This is a generalization of the well-known Boolean Satisfiability Problem (also known as the SAT problem), the first problem that was proven to be NP-complete. With the proliferation of quantum computing, a current approach to tackle this optimization problem is Quantum Annealing (QA). In this work, we compare several gadgets that translate 3-SAT problems into Quadratic Unconstrained Binary Optimization (QUBO) problems to be able to solve them in a quantum annealer. We show the performance superiority of the not-yet-considered gadgets in comparison to state-of-the-art approaches when solving random instances in D-Wave's quantum annealer.

Keywords: Quantum Annealing · Satisfiability · Optimization.

1 Introduction

As the field of Quantum Computing (QC) continues to grow, researchers have been seeking to use quantum properties, such as superposition and entanglement, to help or even surpass the performance of classical computers [2]. One of the main focuses of this field has been to tackle intractable problems that cannot be solved by classical algorithms with running times that grow only polynomially in relation to the input length [9], i.e., problems that are not (or have not been proved to be) in P. A clear example is large number factorization, a problem that Shor's quantum algorithm has theoretically proven to solve in polynomial time, something far away from achieving with classical computers [17]. Another well-known example is the so-called Boolean Satisfiability Problem (SAT, for short). The SAT problem, and in particular, the 3-SAT problem,

⋆ These authors contributed equally to this work.

is of paramount importance in computer science since it is the first problem to be proven NP-complete, meaning that any other NP problem can be translated, i.e., mapped, into a 3-SAT problem [8]. This problem has been tackled recently using Adiabatic Quantic Optimization (AQO) [14], which addresses it by encoding the solution into the ground state of a Hamiltonian and letting the system evolve, relying on the adiabatic theorem of quantum mechanics [9]. More specifically, these optimization tasks have been attended to solve using Quantum Annealing (QA), a technique which can be understood as a heuristic to adiabatic quantum computing [16]. In addition, experimental annealing such as D-Wave experiments, performed in this paper, will not have provable scaling as other algorithms do, and the best which could be shown is an empirically observed scaling, which is unlikely to resolve complexity theory questions since NP-hardness is a worst-case rather than typical-case statement. More on SAT, 3-SAT, and QA will be explained in detail in Section 2.

Our work focuses on implementing and comparing different translations, hereinafter referred to as *gadgets*, to map random 3-SAT instances to Quadratic Unconstrained Binary Optimization (QUBO) problems to be able to solve them using a quantum computer.

The main contributions of our work are:

- We implement new gadgets for translating 3-SAT instances into QUBO formulations and experimentally test them in D-Wave's commercial quantum annealer[4].
- We analyze and compare the characteristics of each gadget, such as the required number of logical and physical qubits, when implementing them in D-Wave's quantum annealer.
- We compare the performance of the new-implemented gadgets in a quantum computer with state-of-the-art gadgets and show their superiority.
- We compare their performance against a classical SAT solver.

This paper is organized as follows. First, in Section 2, we introduce the main concepts relevant to this work. In Section 3, we provide an overview of the relevant research that has contributed to advancements in this domain, highlighting the state-of-the-art gadgets that map 3-SAT to QUBO. Then, in Section 4, we delve into a few gadgets that have been theoretically proposed but have not been implemented yet. Later, in Section 5, we show, compare, and analyze the results obtained from the conducted experiments. Finally, in Section 6, we provide conclusive remarks along with avenues for future exploration.

2 Fundamental Concepts

In this section, we introduce the concepts we will work with, namely the 3-SAT, Max-3-SAT and QUBO problems, and we provide a brief introduction to QA.

[4] https://www.dwavesys.com

2.1 The 3-SAT and Max-3-SAT Problems

The Boolean Satisfiability Problem, also known as SAT for short, is a computational problem in which the goal is to verify if a propositional logic statement formed by boolean variables can be satisfied.

In this work, we focus on 3-SAT, i.e., instances characterized by n boolean variables $\{v_i\}_{i=1}^n$ and a conjunction of m clauses, each one containing the disjunction of 3 literals:

$$F = (l_1 \lor l_2 \lor l_3) \land (l_4 \lor l_5 \lor l_6) \land \ldots \land (l_{3m-2} \lor l_{3m-1} \lor l_{3m}), \tag{1}$$

where every literal l_i refers to a variable v_k or its negation $\neg v_k$.

As stated in Section 1, 3-SAT is of utmost importance since it can provide insights into inherent challenges and strategies involved in solving broader classes of combinatorial optimization and decision problems [8].

On the other hand, Max-SAT (or Max-3-SAT, in our case) is an optimization problem. In this case, the goal is not to verify if the problem is satisfiable but to find an assignment for the variables that satisfy the maximum number of clauses. For 3-SAT, there exists a ratio between the number of variables n and the number of clauses m, $\frac{m}{n} \approx 4.2$, where the generated instance is or is not satisfiable with approximately the same probability. This inflection point is an interesting scenario since, in this situation, the required computational time for classical solvers increases exponentially as the number of variables grows [10]. This work focuses on the number of clauses that are solved for a given instance and not if the instance is itself satisfiable, being the latter a special case of the former. Moreover, we do not expect to get the optimal solution (exactly the maximum number of satisfiable clauses), but a good approximation.[5]

2.2 The QUBO Problem and its Equivalence in Ising Models

In order to solve a SAT problem with a quantum computer, such as a quantum annealer, it is common for the input problem to be formulated in terms of a Binary Quadratic Model (BQM). A typical representation for BQMs are QUBO models, which are characterized by an upper triangular matrix $Q \in \mathbb{R}^{K \times K}$ with K binary variables that take values $x_i = \{0, 1\}$, for i in $0, ..., K-1$. The objective of a QUBO problem is to optimize a function that has the following expression:

$$\min_{\mathbf{x} \in \{0,1\}^K} f(\mathbf{x}), \qquad \text{where } f(\mathbf{x}) = \sum_{0 \le i \le j < K} x_i Q_{ij} x_j + C, \tag{2}$$

$Q_{ij} \in \mathbb{R}$ are the values of the Q matrix, $\mathbf{x} = (x_0, \ldots, x_{K-1})$ refer to the variables of the problem, and C is a constant that it is usually used to shift the optimal output to 0.

[5] Notice that 3-SAT, Max-3-SAT, and even Max-2-SAT are NP-complete problems. Therefore, a bounded-error quantum polynomial-time algorithm for any of these problems would result in a proof of $NP \subseteq BQP$, which, obviously, we do not have.

Analogously to SAT, the solution will be an assignment of $\mathbf{x}$ that minimizes the function $f(\mathbf{x})$.

An equivalent representation of a QUBO problem can be written in terms of an Ising model [11]:

$$\min_{\mathbf{z}\in\{-1,1\}^K} H(\mathbf{z}), \qquad \text{where } H(\mathbf{z}) = -\sum_{0\leq i<j<K} J_{ij} z_i z_j - \sum_i h_i z_i, \tag{3}$$

$J_{ij}, h_i \in \mathbb{R}$ and $z_i = \{-1, 1\}$ for i in $0, ..., K-1$. This formulation plays a prominent role in physics since the variables z_i represent magnetic or quantum spins of a physical system [14].

Via the linear transformation, $x_i = (1 + z_i)/2$ a QUBO problem can be easily mapped to an Ising model, and vice-versa.

In this work, we will work mainly with QUBO problems. However, since the two formulations are equivalent, it is important to recognize that, essentially, we are discussing the same concept when referring to either formulation.

2.3 Quantum Annealing

QA is a quantum computational method to find the global minimum of an objective function. It is a process that occurs in a quantum computer, characterized by a Quantum Processor Unit (QPU). Thus, it relies on quantum bits, commonly referred to as qubits, which stand out for their ability to have their two possible values in superposition, i.e., to be in the ground state (i.e., 0) with probability p and in the excited state (i.e., 1) with probability $1 - p$.

The QPU of a QA has the capability to adjust both the biases of individual qubits and the coupling factors between them, meaning that it can fine-tune the behavior of each qubit and how they interact with one another in order to solve an optimization task. Nevertheless, not all qubits are interconnected via coupling factors; rather, the specific connections depend on the particular QPU architecture.

The experiments conducted in this work are based on D-Wave's *Advantage System 5.4*[6], which uses the *Pegasus* architecture, containing 5000+ qubits and 35000+ coupling factors.

All in all, in order to solve a problem with QA, an objective function needs to be submitted to the QPU, i.e., the values of the bias and coupling factors have to be set. In our case, the diagonal and the upper-diagonal terms of the QUBO matrix Q refer to the biases and coupling factors, respectively.

Once the equivalent physical system of the problem is prepared in the QPU, has a temperature close to absolute zero and it is isolated from the surrounding environment, the system is supposed to work adiabatically and can evolve in time following the quantum mechanics. Consequently, all the amplitudes of the different possible states (i.e., all potential solutions) that started equally will

[6] https://www.dwavesys.com/solutions-and-products/systems/

start to change time-dependently. As a result, the states with higher amplitudes will be the most probable to be present in the system and, in turn, to obtain, once the solution is returned.

As one can expect, the states with higher amplitudes will be those that minimize the objective function submitted to the system. However, as the method is stochastic, i.e., not guaranteed to obtain the most optimum solution, we can obtain either the global minimum of the problem equation or any other state. The results depend on the efficiency of the adiabatic physical system, the defined objective function, the value assignment for the characteristic parameters of the quantum computer, and other errors that can occur during the process. At the end of the QA process, the system will return as a solution the final set of qubits' values and the energy of the system in that state, which ideally will correspond to the solution that minimizes the problem.

3 Related Work

One of the first works that mention the use of gadgets to translate NP-complete problems into Ising formulations is the one from A. Lucas [14]. In this article, the author translates many known problems, such as the knapsack problem, the traveling salesman problem, the graph coloring problem, and the SAT problem, among others, driven by the motivation to solve them via adiabatic quantum optimization algorithms, e.g., quantum annealing. Regarding the satisfiability problem, the author exposes the reduction of 3-SAT to the Maximal Independent Set (MIS) problem, proposed in [6], which is then translated to an Ising model following the procedure in [7].

Nuesslein et al. [15] compare the two main existing translations of 3-SAT to QUBO (i.e., [7] and [5]) and propose two novel approaches to solve instances in the critical region examined by [10]. The authors show theoretically the advantages of their proposed gadgets and achieve better performance when testing them in D-Wave's quantum annealer.

The work of Bian et al. [4] presents a rigorous study on mapping and encoding techniques to solve SAT problems in a quantum annealer and, more specifically, using D-Wave's *Chimera* architecture. However, their proposed translation from 3-SAT to Ising, summarised in Section 4.1, uses a large number of auxiliary variables, being inefficient in our study case. Moreover, their study is not trivially extended to Max-SAT instances.

Finally, Ansotegui & Levy [1] study and propose several new gadgets to tackle the SAT problem using quantum annealers. However, the authors do not show any experimental results nor compare the gadgets to the current state-of-the-art.

Thus, we build our work from [15], completing their comparison by including the new gadgets from [1].

In the following subsections, we expose the gadgets from [15], reviewing the current state-of-the-art and in Section 4, we present the intuition for the gadgets found in [1]. For the sake of simplicity, we are not reporting the results for two of the mappings from max-3-SAT compared in [15], namely the ones proposed in [7] and [5], since they show significantly worse results than the ones presented here. The interested reader can find the results for those methods in the experiments' code repository.

We will show in Section 5, that the gadgets implemented in this work achieve better results compared to the current state-of-the-art.

3.1 Nuesslein^{2n+m}

The first reduction from 3-SAT to QUBO presented in [15], while not optimal, offers valuable insights into some concepts. Following the authors' notation, we will refer to this gadget as *Nuesslein1* from now on.

This gadget requires $2n+m$ qubits, as it allocates two logical qubits for each variable in the problem: one representing the variable and another representing its negation. In addition, it uses a logical qubit for each clause. Note that we explicitly differentiate between logical and physical qubits. The first ones represent the boolean variables of the QUBO problem, and the physical qubits represent the total number of qubits needed to embed the QUBO problem into the machine's particular architecture.

Using 3 auxiliary variables is a useful strategy because the main characteristic of this transcription is to count how many times each variable appears negated or not, along with the relation between variables in each clause. In this way, the importance and weight of each auxiliary variable are easily determined. In summary, with this method, the gadget implicitly assigns the best value $\{True, False\}$ to each variable to satisfy the most number of clauses.

For example, if the variable x_j appears more times negated than not negated, the logical qubit representing the negated variable will have a higher weight than the logical qubit representing the non-negated variable. In this way, the algorithm will try to assign the value *False* to that particular variable.

However, as the authors point out, because of the necessity of using $2n+m$ logical qubits, the number of physical qubits needed to embed the problem is large, making it difficult to obtain optimal results.

3.2 Nuesslein^{n+m}

The second approach presented in [15] is a reduction from 3-SAT to QUBO using only $n+m$ logical variables, i.e., it uses a logical qubit per variable and another logical qubit per clause. Following the authors' notation, we will refer to this gadget as *Nuesslein2* from now on.

The idea behind this gadget is to update the QUBO matrix by recursively scanning each clause individually. In essence, the algorithm first sorts all clauses depending on the number of negated literals appearing in each one, obtaining an expression similar to the following:

$$(a \lor b \lor c), (a \lor b \lor \neg c), (a \lor \neg b \lor \neg c), (\neg a \lor \neg b \lor \neg c). \tag{4}$$

The goal is to construct matrices that depict the relationships between literals and their existing connections, aiming to determine how each clause pattern influences the QUBO matrix.

The usefulness of determining relations among literals and other associations allows the gadget to identify patterns and, when possible, merge two clauses into one.

For instance, if we are presented with the following pair of clauses: $(v_1 \vee v_2 \vee v_3), (v_1 \vee v_2 \vee v_4)$ the gadget can spot that the relation $(v_1 \vee v_2)$ is repeated, merging the two clauses into only one: $(v_4 \vee (v_1 \vee v_2) \vee v_3)$. As a result, and as opposed to the previous gadget, one auxiliary logic qubit is no longer needed.

4 New Approaches

In this section, we expose two additional transcriptions to reduce 3-SAT to QUBO, extracted from [1].

Both approaches require $n+m$ logical qubits. That is, they require a logical variable x_i, for each variable v_i, for $i = 1, \dots n$, and another logical variable b_j, for each clause c_j, for $j = 1, \dots m$. Despite using the same number of logical variables than *Nuesslein2*, superior experimental results will be shown in Section 5.

4.1 CJ1^{n+m}

In [4], a "divide-and-conquer" approach is presented for transforming a SAT formula into an Ising problem. The general procedure, detailed in [4], follows three steps:

1. Factorize the input formula rewriting every conjunct that is not small enough to be easily mapped to Ising. Each of these large conjuncts should be transformed, by means of Tseitin transformation [18], into an equivalently-satisfiable formula where each conjunct can be easily mapped to Ising. As a result of this step, our formula will be a conjunction of subformulas, with each subformula easily mapped to Ising.
2. Independently map each of the subformulas to Ising. This means that new variable replicas should be created for each subformula.
3. For each different variable, link their variable replicas by means of a chain of equivalences.

However, this approach requires a large number of logical variables. In particular, it requires $n + 2m$ logical variables plus those needed to fulfill the last step of the algorithm (i.e., to link each variable's replicas).

As an alternative, Ansotsegui & Levy theoretically develop a similar approach in [1], which benefits from the Tseitin encoding but requires only $n + m$ logical variables. We label this gadget as *CJ1*.

The intuition behind this method is to transcript each clause independently, modifying iteratively the values of the QUBO matrix $Q \in \mathbb{R}^{K \times K}$. Since each clause is translated independently, we will only refer to the j'th 3-SAT clause as a reference to this explanation.

The reduction from 3-SAT starts by implementing the Tseitin encoding in [18], obtaining the following expressions:

$$l_1 \vee l_2 \vee l_3 \rightarrow \begin{cases} l_1 \vee l_2 \leftrightarrow b_j \\ b_j \vee l_3, \end{cases} \tag{5}$$

where l_i represents the variable in the i'th literal of the clause, and b_j represents the auxiliary variable added in the j'th clause.

The QUBO matrix can now be computed directly by expressing (5) in terms of the contribution to the objective function (2).

As explained in [1], each of the expressions in 5 contributes to an Ising model in the following way:

$$l_1 \vee l_2 \leftrightarrow b_j \quad \Leftrightarrow \quad H_1(\mathbf{x'}, \mathbf{b'}) = \frac{5}{2} + \frac{1}{2}x'_1 + \frac{1}{2}x'_2 - b'_j + \frac{1}{2}x'_1x'_2 - x'_1b'_j - x'_2b'_j \tag{6}$$

$$b_j \vee l_3 \quad \Leftrightarrow \quad H_2(\mathbf{x'}, \mathbf{b'}) = \frac{1}{2} - \frac{1}{2}b'_j - \frac{1}{2}x'_3 + \frac{1}{2}x'_3b'_j \tag{7}$$

where $x'_i = s_i\sigma_i$, being $s_i = -1$ $(+1)$ the (not) negation of the literal l_i, and $\sigma_i = \{-1, 1\}$ the value assigned to that literal. The final contribution for the entire gadget will be the sum of both terms.

As stated in Section 2, an Ising model can be easily mapped to a QUBO problem. Using a variable transformation such that $\sigma_i = 2x_i - 1$ and $b'_j = 2b_j - 1$, the final expression that contributes to the QUBO objective function is:

$$\begin{aligned} f(\mathbf{x}, \mathbf{b}) =& (3s_1 - s_1s_2)x_1 + (3s_2 - s_1s_2)x_2 - 2s_3x_3 - (3 + 2s_1 + 2s_2 - s_3)b_j \\ &+ 2s_1s_2x_1x_2 - 4s_1x_1b_j - 4s_2x_2b_j + 2s_3x_3b_j + C, \end{aligned} \tag{8}$$

where C is a constant.

The values that contribute to each term of the matrix $Q \in \mathbb{R}^{L \times L}$ are the coefficients of 8. For instance, $Q_{l_1l_1} = 3s_1 - s_1s_2$, and $Q_{l_3b_j} = 2s_3$. Thus, all contributions for each clause will be summed into the matrix.

This approach stands out for the couple between the two first literals in the clause, while the third literal only couples with the auxiliary variable. This is important when working in a QPU, in which the architecture and the available connections between qubits are determining factors in the results obtained.

4.2 CJ2^{n+m}

The second approach presented in [1] follows a similar implementation, also focusing on transcribing each clause individually. However, this gadget's first reduction is from 3-SAT to Max-2-SAT, that is, to a conjunction of clauses involving only 2 variables.

Analogously to what is done in Section 4.1, the expression obtained for the j'th clause using this gadget is:

$$l_1 \lor l_2 \lor l_3 \rightarrow \begin{cases} l_1 \lor l_3, \neg l_1 \lor \neg l_3 \\ l_1 \lor \neg b_j, \neg l_1 \lor b_j \\ l_3 \lor \neg b_j, \neg l_3 \lor b_j \\ l_2 \lor b_j \end{cases} \tag{9}$$

As before, the authors show the intuition to express 9 into contributions to an Ising model:

$$H(\mathbf{x'}, \mathbf{b'}) = \frac{1}{2}(-x'_2 - b'_j + x'_1x'_3 - x'_1b'_j - x'_3b'_j + x'_2b'_j) \tag{10}$$

Applying the same transformation as in the previous section, we obtain that the contribution to the QUBO objective function is:

$$\begin{aligned} f(\mathbf{x}, \mathbf{b}) =& (s_1 - s_1s_3)x_1 - 2s_2x_2 + (s_3 - s_1s_3)x_3 - (1 - s_1 + s_2s_3)b_j \\ &+ (2s_1s_3)x_1x_3 - 2s_1x_1b_j + 2s_2x_2b_j - 2s_3x_3b_j + C, \end{aligned} \tag{11}$$

where C is a constant.

The values for $Q_{L_iL_i}$ of the QUBO matrix will be the sum of all the coefficients in 11 for each variable x_i and b_j.

Similar to the previous gadget, this approach only adds one auxiliary variable for each clause, obtaining a total of $n + m$ logical variables.

5 Experimental Results

Following the procedure used in [15], we have generated random 3-SAT instances in the critical region, i.e., where the ratio of the number of clauses to the number of variables approaches $\frac{m}{n} \approx 4.2$ [10]. In particular, we have generated 20 random instances for 5 different scenarios: $n = 5$ and $m = 21$, $n = 10$ and $m = 42$, $n = 12$ and $m = 50$, $n = 20$ and $m = 84$ and for $n = 50$ and $m = 210$.

We let D-Wave find an appropriate embedding automatically for each instance and generated 100 shots, i.e., the algorithm returned 100 possible solutions. Moreover, we set 100 μs as the annealing time in each solution, and we used the parameter *reduce_ intersample_ correlation* to reduce sample-to-sample correlations. The values for the number of shots and the annealing time have been chosen according to the results in [12], where they benchmark these two parameters for different sizes of QUBO problems using D-Wave's architecture.

Analogously to [15], we present in Fig.1 the number of non-zero elements existent in the corresponding QUBO matrices. Although the *Nuesslein1*, *CJ1*, and *CJ2* approaches utilize the same number of logical variables, the assignment of each coupling factor (Q_{ij}, $i < j$) differs from one another. Thus, it can be seen how this results in a slight enhancement in the number of non-zero couplings for

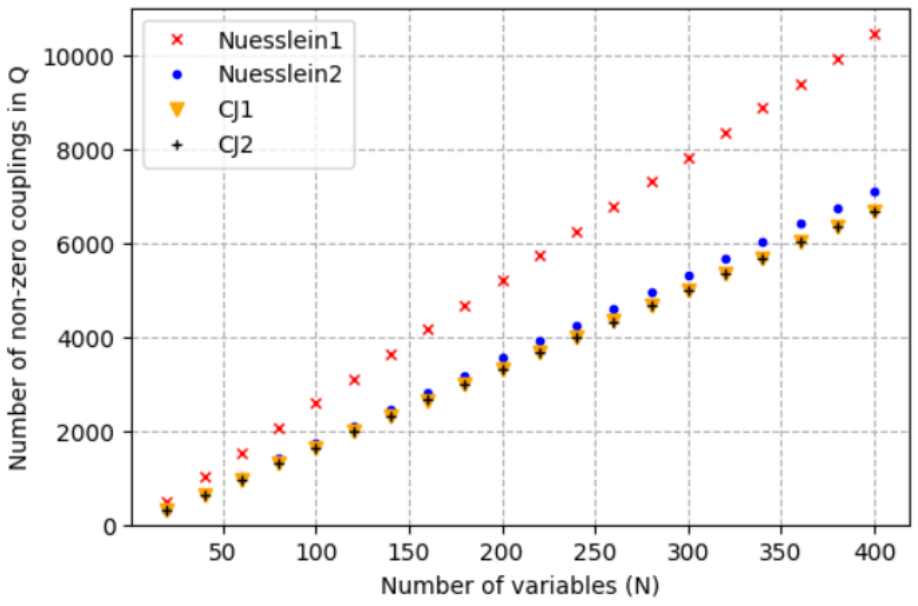


Fig. 1: Number of non-zero couplings in the QUBO matrix for 20 different 3-SAT instances varying the number of variables (N). Variance is negligibly discernible to the naked eye.

the *CJ1* and *CJ2* gadgets. In other words, these gadgets are able to construct the QUBO matrix with fewer elements.

On the other hand, the number of physical qubits depending on the instance's number of variables is essentially equivalent for *Nuesslein2*, *CJ1*, and *CJ2*. As mentioned in [15], *Nuesslein1* exhibits poorer results in this experiment. For a detailed comparison, readers are directed to consult Table 2 in Appendix A.

In Table 1, we present the mean number of satisfied clauses achieved with the best-found variable assignment for each scenario and each gadget. The last row of Table 1 displays the outcomes obtained by the *MaxSatZ* exact solver [13], which finds always the optimal solution.

We can see from Table 1 that we recover the results presented in [15]; that is, Nuesslein's second gadget performs better than their first gadget. Nevertheless, the two gadgets reviewed in this work perform better than Nuesslein's second gadget in all scenarios. For those problems with fewer variables, the *CJ1* gadget performs slightly better, while for bigger problems, the *CJ2* gadget exhibits the best performance.

Although the trait of non-zero couplings present in the QUBO matrix could suggest that the *CJ1* and the *CJ2* approaches may scale and demonstrate superior performance as the problem size expands, we think that the difference depicted in Fig.1 is not sufficient to account for the enhanced performance observed in the newly implemented gadgets.

Table 1: Experimental results obtained i) by using the analyzed gadgets and D-Wave's quantum annealer (rows 1-4) and ii) an exact SAT solver (row 5). The numbers quoted represent the means of satisfied clauses for all different instances in each scenario.

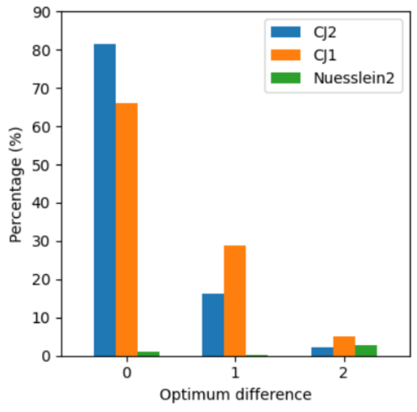
	n=5, m=21	n=10, m=42	n=12, m=50	n=20, m=84	n=50, m=210
Nuesslein1	18.35	37.3	43.95	73.2	183.4
Nuesslein2	20.65	41.4	49.3	82.6	204.55
CJ1	**20.7**	**41.7**	**49.7**	83.05	204.25
CJ2	**20.7**	41.65	49.65	**83.25**	**207.0**
MaxSatZ	20.7	41.75	49.8	83.6	209.65

Moreover, since the number of logical qubits is the same for the three top-performing gadgets (i.e., *Nuesslein2*, *CJ1* and *CJ2*), and there is no notable difference in the use of physical qubits when embedding the problems into the QPU, we conclude that some other inner characteristic of the gadgets must play a crucial role. In this sense, we strongly believe that the energy gap of each gadget, i.e., the energy difference between the ground state of the system and the first excited state, or in other words, the difference between the optimal output value and the subsequent suboptimal output value, must be of paramount importance.

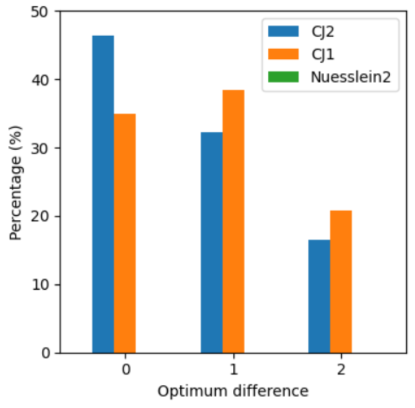
Fig.2 provides a new insight for assessing the efficiency of the analyzed gadgets. Each bar in the histogram represents the percentage of shots, i.e., the percentage of sample solutions that found the optimal or subsequent suboptimal solutions for each gadget and problem scenario. Thus, the x-axis comprises three labels: "0", "1", and "2", which denote whether the solution found by a given shot was the optimal, the first suboptimal, or the subsequent second suboptimal, compared to the exact solver.

For instance, from Fig.2a, it can be inferred that when using the *CJ2* gadget within the instances composed of $n = 5$ variables, the optimal solution was found approximately 80% of the time. In other words, around 1600 shots found the optimal solution out of a total of $20 \cdot 100 = 2000$ shots. In addition, the subsequent suboptimal, that is, the solution that satisfied the most clauses but not as many as in the optimal case, was found around 20% of the time. As opposed to this, one can also infer that the *CJ1* gadget obtained less optimal solutions within this context. Note that *Nuesslein2* does not practically appear in the plots, being the worst approach in this case. Based on the number of samples and the results in Fig. 2 we believe that the results are statistically significant.

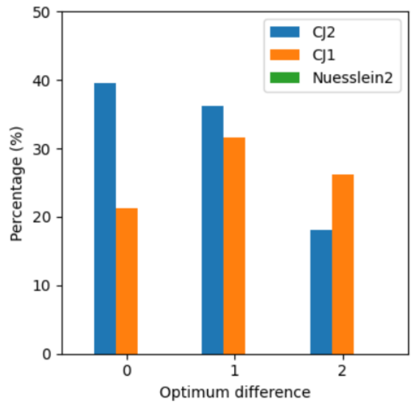
From Fig.2, it can be concluded that *CJ2* seems to be the best approach when solving (Max-)3-SAT instances in D-Wave's quantum computer since it is the gadget that more frequently finds the optimal solution. Moreover, it can also be inferred that *CJ2* exhibits superior scalability relative to its counterparts, making it the best option for dealing with this type of problem.



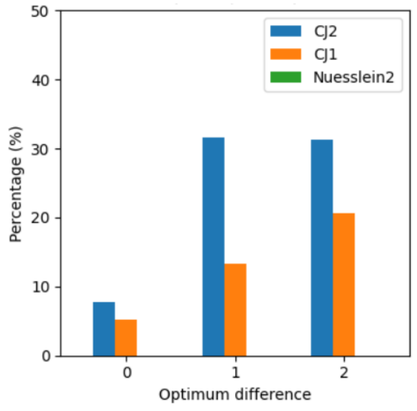
(a) Instances with n=5 and m=21.



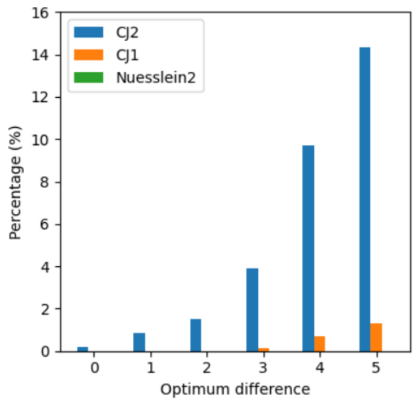
(b) Instances with n=10 and m=42.



(c) Instances with n=12 and m=50.



(d) Instances with n=20 and m=84.



(e) Instances with n=50 and m=210.

Fig. 2: Histogram representing the percentage of shots that find the optimal or subsequent suboptimal solutions for each gadget and problem scenario. The x-axis indicates the quality of the solution found, i.e., 0 indicates the optimal solution, 1 indicates the next suboptimal solution, and so on and so forth. Each triplet of bars in each subfigure depicts, from left to right, the *CJ2*, *CJ1*, and *Nuesslein2* approach.

A repository with all the analyses and experiments can be found in *https://github.com/IIIA-ML/QuantumSAT.git*.

6 Conclusion and Future Work

Motivated to solve NP problems using quantum computers and constructing our work completing Nuesslein et al.'s prior research [15], we have implemented new gadgets to reduce 3-SAT instances to QUBO formulations. We have empirically tested the performance using D-Wave's quantum annealer and have outperformed the current state-of-the-art approaches.

The immediate subsequent task should entail comprehending the reasons behind the superior performance of these results compared to the state-of-the-art approaches. In [3] it is mentioned that, in general, larger energy gaps lead to higher success rates in the QA process. A theoretical examination of the energy gap among the gadgets could be conducted under the same parameter conditions (e.g. the range of the QUBO values when developing the gadget), ensuring a fair and equitable comparison. It is, however, non-trivial to normalize all research in a unique family of topology parameters.

It also remains a future challenge to deal with the embedding of bigger problems into the current architecture to further test the scalability and robustness of our methods and benchmark the capabilities of the current QPUs. Furthermore, we seek to extend our investigations into real-world applications by exploring industrial instances or those sourced from SAT competitions, enhancing the practical relevance of our findings.

Along these lines, it would also be promising to delve into more complex problem domains, such as k-SAT with $k > 3$, broadening the scope of our research to encompass a wider range of challenges.

Acknowledgments. This work was founded partially by the Government of Catalonia Exp. Num. 2023 DI 00071, by Enzyme Advising Group under the development of an industrial PhD on Quantum Machine Learning, by CSIC's JAE Intro JAE-ICU_23_00782 and by grant PID2022-136787NB-I00 funded by MCIN/AEI/10.13039/501100011033. Jesus Cerquides is funded by European Union KHealthInAir, GUARDEN, and Humane-AI projects with No. 101057693, 101060693, and 952026.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Ansótegui, C., Levy, J.: SAT, Gadgets, Max2XOR, and Quantum Annealers (Feb 2024), `http://arxiv.org/abs/2403.00182`, arXiv:2403.00182 [quant-ph]
2. Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., Lloyd, S.: Quantum machine learning. Nature **549**(7671), 195–202 (Sep 2017). `https://doi.org/10.1038/nature23474`, `https://www.nature.com/articles/nature23474`, number: 7671 Publisher: Nature Publishing Group

3. Bian, Z., Chudak, F., Israel, R., Lackey, B., Macready, W.G., Roy, A.: Discrete optimization using quantum annealing on sparse Ising models. Frontiers in Physics **2** (2014), https://www.frontiersin.org/articles/10.3389/fphy.2014.00056
4. Bian, Z., Chudak, F., Macready, W., Roy, A., Sebastiani, R., Varotti, S.: Solving SAT (and MaxSAT) with a quantum annealer: Foundations, encodings, and preliminary results. Information and Computation **275**, 104609 (Dec 2020). https://doi.org/10.1016/j.ic.2020.104609, https://www.sciencedirect.com/science/article/pii/S0890540120300973
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. Scientific Reports **6**(1), 37107 (Nov 2016). https://doi.org/10.1038/srep37107, https://www.nature.com/articles/srep37107, number: 1 Publisher: Nature Publishing Group
6. Choi, V.: Adiabatic Quantum Algorithms for the NP-Complete Maximum-Weight Independent Set, Exact Cover and 3SAT Problems. ArXiv (Apr 2010), https://www.semanticscholar.org/paper/Adiabatic-Quantum-Algorithms-for-the-NP-Complete-Choi/1fcb4d5749074714ca2eb0b56ab986ca66fd0b95
7. Choi, V.: Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. Quantum Information Processing **7**(5), 193–209 (Oct 2008). https://doi.org/10.1007/s11128-008-0082-9, https://doi.org/10.1007/s11128-008-0082-9
8. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the third annual ACM symposium on Theory of computing. pp. 151–158. STOC '71, Association for Computing Machinery, New York, NY, USA (1971). https://doi.org/10.1145/800157.805047, https://dl.acm.org/doi/10.1145/800157.805047
9. Farhi, E., Goldstone, J., Gutmann, S., Lapan, J., Lundgren, A., Preda, D.: A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem. Science **292**(5516), 472–475 (Apr 2001). https://doi.org/10.1126/science.1057726, http://arxiv.org/abs/quant-ph/0104129, arXiv:quant-ph/0104129
10. 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 (Feb 2019), http://arxiv.org/abs/1902.04703, arXiv:1902.04703 [quant-ph] version: 1
11. Glover, F., Kochenberger, G., Du, Y.: A Tutorial on Formulating and Using QUBO Models (Nov 2019). https://doi.org/10.48550/arXiv.1811.11538, http://arxiv.org/abs/1811.11538, arXiv:1811.11538 [quant-ph]
12. Gonzalez Calaza, Carlos D., W.D.M.K.: Garden optimization problems for benchmarking quantum annealers. Quantum Information Processing (2021). https://doi.org/10.1007/s11128-021-03226-6, https://doi.org/10.1007/s11128-021-03226-6
13. Li, C.M.: Detecting Disjoint Inconsistent Subformulas for Computing Lower Bounds for Max- SAT
14. Lucas, A.: Ising formulations of many NP problems. Frontiers in Physics **2** (2014). https://doi.org/10.3389/fphy.2014.00005, http://arxiv.org/abs/1302.5843, arXiv:1302.5843 [cond-mat, physics:quant-ph]
15. Nüßlein, J., Zielinski, S., Gabor, T., Linnhoff-Popien, C., Feld, S.: Solving (Max) 3-SAT via Quadratic Unconstrained Binary Optimization (Feb

2023). https://doi.org/10.48550/arXiv.2302.03536, http://arxiv.org/abs/2302.03536, arXiv:2302.03536 [quant-ph]

16. Schuld, M., Petruccione, F.: Supervised Learning with Quantum Computers. Quantum Science and Technology, Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-96424-9, https://link.springer.com/10.1007/978-3-319-96424-9
17. Shor, P.W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Journal on Computing **26**(5), 1484–1509 (Oct 1997). https://doi.org/10.1137/S0097539795293172, https://epubs.siam.org/doi/10.1137/S0097539795293172, publisher: Society for Industrial and Applied Mathematics
18. Tseitin, G.S.: On the Complexity of Derivation in Propositional Calculus. In: Siekmann, J.H., Wrightson, G. (eds.) Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970, pp. 466–483. Symbolic Computation, Springer, Berlin, Heidelberg (1983). https://doi.org/10.1007/978-3-642-81955-1_28

A Number of physical qubits required per gadget

Table 2: Mean values and standard deviations of the required number of physical qubits for each gadget and each scenario.

	n=15, m=63	n=18, m=75	n=21, m=88	n=24, m=100	n=27, m=113
Nuesslein1	238.90 ± 13.6	308.40 ± 20.6	392.00 ± 22.2	466.90 ± 23.7	561.40 ± 24.7
Nuesslein2	150.15 ± 6.0	196.55 ± 11.7	239.75 ± 10.8	282.40 ± 11.5	342.80 ± 15.5
CJ1	147.25 ± 4.9	185.95 ± 11.6	234.05 ± 13.5	277.90 ± 14.4	332.05 ± 21.5
CJ2	145.75 ± 8.0	187.85 ± 8.6	234.50 ± 10.9	280.55 ± 8.9	331.10 ± 13.9