Using an Evolutionary Algorithm to Create (MAX)-3SAT QUBOs

Zielinski, Sebastian; Zorn, Maximilian; Gabor, Thomas; Feld, Sebastian; Linnhoff-Popien, Claudia

DOI
10.1145/3638530.3664153

Publication date
2024

Document Version
Final published version

Published in
GECCO '24 Companion

Using an Evolutionary Algorithm to Create (MAX)-3SAT QUBOs

Sebastian Zielinski
Institute for Informatics, LMU Munich
Munich, Germany
sebastian.zielinski@ifi.lmu.de

Maximilian Zorn
Institute for Informatics, LMU Munich
Munich, Germany
maximilian.zorn@ifi.lmu.de

Thomas Gabor
Institute for Informatics, LMU Munich
Munich, Germany
thomas.gabor@ifi.lmu.de

Sebastian Feld
Quantum & Computer Engineering,
Delft University of Technology
Delft, The Netherlands
s.feld@tudelft.nl

Claudia Linnhoff-Popien
Institute for Informatics, LMU Munich
Munich, Germany
linnhoff@ifi.lmu.de

ABSTRACT

A common way of solving satisfiability instances with quantum methods is to transform these instances into instances of QUBO. State-of-the-art transformations from MAX-3SAT to QUBO work by mapping clauses of a 3SAT formula associated with the MAX-3SAT instance to an instance of QUBO and combining the resulting QUBOs into a single QUBO instance representing the whole MAX-3SAT instance. As creating these transformations is currently done manually or via exhaustive search methods and is, therefore, algorithmically inefficient, we see potential for including search-based optimization. In this paper, we propose two methods of using evolutionary algorithms to create QUBO representations of MAX-3SAT problems automatically. We evaluate our created QUBOs on 500 and 1000-clause 3SAT formulae and find competitive performance to state-of-the-art baselines when using both classical and quantum annealing solvers.

CCS CONCEPTS

• **Hardware → Quantum computation**; • **Mathematics of computing → Combinatorial optimization**; • **Theory of computation → Theory of randomized search heuristics**.

KEYWORDS

QUBO, (MAX)-3SAT, combinatorial optimization, evolutionary algorithm

ACM Reference Format:
Sebastian Zielinski, Maximilian Zorn, Thomas Gabor, Sebastian Feld, and Claudia Linnhoff-Popien. 2024. Using an Evolutionary Algorithm to Create (MAX)-3SAT QUBOs. In *Genetic and Evolutionary Computation Conference (GECCO '24 Companion), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3638530.3664153

GECCO '24 Companion, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0495-6/24/07
https://doi.org/10.1145/3638530.3664153

1 INTRODUCTION

Over the past decade, Quadratic Unconstrained Binary Optimization (QUBO) has become a unifying framework for modeling combinatorial optimization problems [40]. A significant portion of the growing interest in QUBO can be attributed to developments in the field of quantum computing, where QUBO serves as an input format for quantum annealers and the quantum approximate optimization algorithm on gate-based quantum computers. As a consequence of the increased availability and problem-solving capabilities of quantum hardware systems, researchers are investigating methods of transforming problems from various domains to instances of QUBO, to be able to evaluate the problem-solving capabilities of contemporary quantum hardware systems for their specific problems. Examples of this effort include applications in finance, [24, 38], logistics [11, 21], scheduling [18, 39], and many more [26].

In this paper we are concerned with creating new methods for finding QUBO formulations for satisfiability problems. Given a formula of propositional logic, the satisfiability problem (SAT) asks whether an assignment of Boolean values to the variables of the given formula exists such that the formula is satisfied. Satisfiability problems are ubiquitous in computer science. In theoretical computer science, they are often used to prove the NP-hardness of other problems. In practice, they occur in many application domains like circuit design and verification [28], dependency resolution [1], and planning [35]. As every satisfiability problem can efficiently be reduced to the case where each clause of the satisfiability problem consists of at most three variables (3SAT), we will focus on 3SAT problems in this paper. The optimization version of a 3SAT problem is a MAX-3SAT problem. In a MAX-3SAT problem, the goal is to find an assignment to the variables of a 3SAT formula such that as many clauses as possible are satisfied. To transform MAX-3SAT instances into instances of QUBO, researchers developed several different methods (e.g., [7, 9, 32]). It has been shown that the choice of a QUBO transformation for a given MAX-3SAT instance can significantly impact the quality of the solutions when solving these problems on a quantum annealer [23, 44]. Thus, creating QUBO formulations that yield better results when solving MAX-3SAT problems on a quantum annealer can be seen as an effort to bring this widely studied method closer to practical application.

In this paper, we propose two methods for using evolutionary algorithms to create QUBO formulations for MAX-3SAT problems. To the best of our knowledge, evolutionary algorithms have not yet

been used to create QUBO formulations for specific instances of combinatorial optimization problems. We thus consider this work to be of explorative nature to show the feasibility of this approach in general. Our contributions in this paper are:

1. According to the concept of Pattern QUBOs [32, 45], which will be revisited in Sec. 2.3, to create a QUBO representation of a MAX-3SAT instance it suffices to transform each clause of the 3SAT formula of the MAX-3SAT instance into an instance of QUBO and then combine all resulting QUBO instances. Currently, two methods of finding transformations from 3SAT clauses to QUBO exist. The first one is that a scientist manually creates this transformation; the second method uses the Pattern QUBO approach [45], which employs an exhaustive search. As our first contribution, we propose an evolutionary algorithm that automatically finds QUBO representations for 3SAT clauses. Furthermore, we show how to adapt this method to create QUBO representations for 3SAT clauses that fulfill certain design criteria (i.e., a desired sparsity, a specific gap between the optimal energies and the energies of incorrect solutions, etc.). Thus, we enable the automatic generation of specific QUBO representations for 3SAT clauses without performing any procedure based on an exhaustive search nor a manual creation process.
2. In current state-of-the-art MAX-3SAT to QUBO transformations that are based on the Pattern QUBO concept, each clause of the 3SAT formula associated with the MAX-3SAT problem is assigned one of the *clause types* $\{0, 1, 2, 3\}$. The clause type is the count of negated variables within the clause (e.g., a clause of type 0 possesses zero negated variables). The transformation then provides a single method of transforming each clause of a given type ($\{0, 1, 2, 3\}$) to an instance of QUBO. Thus, all clauses of a given type (e.g., all clauses with zero negated variables) will be transformed into an instance of QUBO according to a single transformation rule.
   As our second contribution, we propose an evolutionary algorithm that can choose an individual QUBO transformation rule for each of the clauses of a 3SAT formula, instead of using a single transformation rule for all clauses of the same type. As the solution landscape of a QUBO representation of a MAX-3SAT problem is induced by the choices of the QUBO transformations for the 3SAT clauses associated with the MAX-3SAT problem, our approach enables the creation of QUBO representations with many different solution landscapes. The goal of our second evolutionary method is thus to find choices of QUBO transformation rules for clauses of a 3SAT formula such that the resulting solution landscape is beneficial for a given optimizer.

The remainder of this paper is organized as follows: In Sec. 2, we introduce the necessary foundations of satisfiability problems as well as population-based optimization and give a detailed explanation of the Pattern QUBO creation process. In Sec. 3 we discuss related work on satisfiability as well as thematically related quantum optimization. We then formalize our evolutionary approach in Sec. 4, for the QUBO search and selection respectively, and discuss our experimental evaluation in the same order in Sec. 5. Finally, we conclude our findings in Sec. 6.

2 FOUNDATIONS

2.1 Satisfiability Problems

Satisfiability problems are concerned with the satisfiability of Boolean formulae. Thus, we first define a Boolean formula:

Definition 1 (Boolean formula [5]). *A* Boolean formula *consists of the Boolean variables $x_1, ..., x_n$ and the logical operators $\wedge$, $\vee$, $\neg$. A vector $z \in \{0,1\}^n$ of Boolean values (we identify the value 1 as TRUE and the value 0 as FALSE) is called an* assignment *as it assigns truth values to the Boolean variables $x_1, ..., x_n$ as follows: $x_i := z_i$, where z_i is the ith component of z. For a Boolean formula ϕ and an assignment $z \in \{0,1\}^n$, we call $\phi(z)$ the evaluation of ϕ when the variable x_i is assigned the Boolean value z_i. If there exists a $z \in \{0,1\}^n$ such that $\phi(z)$ is TRUE, we call ϕ satisfiable. Otherwise, we call ϕ unsatisfiable.*

Satisfiability problems are often given in conjunctive normal form, which we define next:

Definition 2 (Conjunctive Normal Form [5]). *A Boolean formula over variables $x_1, ..., x_n$ is in* Conjunctive Normal Form (CNF) *if it is of the following structure:*

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

Each y_{i_j} is either a variable x_k or its negation $\neg x_k$. The y_{i_j} are called the literals *of the formula. The terms $(\vee_j y_{i_j})$ are called the* clauses *of the formula. A k*CNF *formula is a CNF formula in which all clauses contain at most k literals.*

Given a Boolean formula ϕ in *kCNF*, the satisfiability problem is the task of determining whether ϕ is satisfiable or not. This problem was one of the first problems for which NP-completeness has been shown [10]. In this paper, we will especially consider 3*CNF* satisfiability problems, to which we will refer as 3SAT problems.

The optimization version of a satisfiability problem is the MAX-SAT problem. In the MAX-SAT problem, we are given a Boolean formula ϕ consisting of m clauses. The task is to find an assignment of truth values to the variables of ϕ such that as many clauses as possible are satisfied. Finding an assignment in the MAX-SAT problem that satisfies m clauses is thus equivalent to solving the corresponding satisfiability problem (i.e., deciding whether ϕ is satisfiable or not). MAX-SAT is thus NP-hard as well. To avoid confusion, we want to emphasize that a MAX-3SAT instance consists of a 3SAT formula (and not a "MAX-3SAT formula," which does not exist as a notion), for which the maximum number of satisfiable clauses should be determined.

2.2 Quadratic Unconstrained Binary Optimization (QUBO)

In this section we will formally introduce quadratic unconstrained binary optimization (QUBO) and related terminology that will be used in the remainder of this paper.

Definition 3 (QUBO [16]). *Let $Q \in \mathbb{R}^{n \times n}$ be a square matrix and let $x \in \{0,1\}^n$ be an n-dimensional vector of Boolean variables. The QUBO problem is defined as follows:*

$$\textit{minimize} \quad H_{QUBO}(x) = x^T Q x = \sum_i Q_{ii} x_i + \sum_{i<j} Q_{ij} x_i x_j \qquad (1)$$

We call $H_{QUBO}(x)$ the (QUBO) energy of vector x. The matrix Q will also be called *QUBO matrix*. Representing a QUBO matrix as an upper triangular matrix is customary.

In this paper, we will create QUBO instances that contain auxiliary variables. That means that, to transform a given MAX-3SAT instance to an instance of QUBO, we introduce additional variables that do not correspond to any variables of the given MAX-3SAT instance. We will often say that an assignment $\vec{x} = (x_1 := v_0, \ldots, x_n := v_n)$ of Boolean values $v_i \in \{0, 1\}$ to the variables $x_1, \ldots, x_n$ of the MAX-3SAT instance has energy E in Q, by which we mean:

$$min\ \{(\vec{x}, y)^T Q(\vec{x}, y) \mid y \in \{0, 1\}^m\} = E \tag{2}$$

Here Q is a QUBO matrix and $(\vec{x}, y)$ is an $(n + m)$-dimensional column vector defined as $(\vec{x}, y) = (x_1 = v_0, \ldots, x_n = v_n, y_1, \ldots, y_m)$. The first n values of the vector $(\vec{x}, y)$ are given by the assignment $\vec{x} = (x_1 := v_0, \ldots, x_n := v_n)$ of Boolean values $v_i \in \{0, 1\}$ to the variables $x_1, \ldots, x_n$ of the MAX-3SAT instance. The last m entries represent the values of the auxiliary variables $y_1, \ldots, y_m$.

2.3 Pattern QUBOs

In this section, we will review the concept of Pattern QUBOs, as introduced in [32, 45]. Let ϕ be a 3SAT formula consisting of m clauses. For each clause, we sort the variables such that all negated variables are always at the end of the clause. For example, the sorted version of the clause $(x_1 \vee \neg x_2 \vee x_3)$ is the clause $(x_1 \vee x_3 \vee \neg x_2)$. As a consequence of this sorting procedure, there are now only four different types of 3SAT clauses in a 3SAT instance:

Type 0 — no negations: $(a \vee b \vee c)$
Type 1 — one negation: $(a \vee b \vee \neg c)$
Type 2 — two negations: $(a \vee \neg b \vee \neg c)$
Type 3 — three negations: $(\neg a \vee \neg b \vee \neg c)$

The idea of Pattern QUBOs is to find QUBO representations for each of these four types of clauses such that assignments that satisfy the clause correspond to the minimum in the QUBO representation. Likewise, assignments that do not satisfy a clause (of which there is only one for each clause) correspond to a non-minimal higher value in the QUBO representation. We apply the following procedure to transform any clause of ϕ to an instance of QUBO. First, we assess the type of the respective clause (i.e., we count the number of negated variables). Next, we sort the clause so that all negated variables are at the end of the clause. Finally, we replace all the variables of the Pattern QUBO of the same type as our current clause with the variables of the current clause (hence the name "Pattern QUBOs"). After applying this procedure for each of the m clauses of ϕ, we combine the resulting m QUBO representations of the clauses into a single QUBO representation.

To illustrate this procedure, suppose we are given the formula $\phi = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$ and the Pattern QUBO for a type 0 clause shown in Tab. 1.

The variable y in the QUBO shown in Tab. 1 is an auxiliary variable that is needed to be able to guarantee that each satisfying assignment of the clause $(a \vee b \vee c)$ has minimum energy in the QUBO, while each non-satisfying assignment (which is only $a = b = c = 0$) has a higher energy in the QUBO (see Sec. 2.2). To transform ϕ into an instance of QUBO we apply the procedure described above for each of the clauses of ϕ. First, we observe that

Table 1: Pattern QUBO for a type 0 clause $(a \vee b \vee c)$

	a	b	c	y
a		2		-2
b				-2
c			-1	1
y				1

both clauses of ϕ do not contain any negations. Hence, both clauses are of type 0. As there are no negations, we do not have to sort the variables of any of the clauses. As our next step, we replace the variables of the Pattern QUBO shown in Tab. 1 with the variables of the first (or second, respectively) clause of ϕ. The result of this step is shown in Tab. 2.

Table 2: Applied Pattern QUBOs for given formula ϕ

(a) $(x_1 \vee x_2 \vee x_3)$

	x_1	x_2	x_3	y_1
x_1		2		-2
x_2				-2
x_3			-1	1
y_1				1

(b) $(x_1 \vee x_2 \vee x_4)$

	x_1	x_2	x_4	y_2
x_1		2		-2
x_2				-2
x_4			-1	1
y_1				1

In our last step, we now combine the two QUBO matrices shown in Tab. 2 to receive a QUBO representation of ϕ.

Note that, when minimizing a QUBO problem with QUBO matrix Q, we want to find the minimum of $\sum_i Q_{ii}x_i + \sum_{i<j} Q_{ij}x_ix_j$ (see Eq. 1). Thus, in the case of the QUBO shown in Tab. 2(a) we want to find an assignment of Boolean values to the variables x_1, x_2, x_3, y_1 such that $P_1 = -x_3 + y_1 + 2x_1x_2 - 2x_1y_1 - x_2y_2 + x_3y_1$ is minimized. Equivalently, in the case of the QUBO shown in Tab. 2(b) we want to find an assignment of Boolean values to the variables x_1, x_2, x_4, y_2 such that $P_2 = -x_4 + y_2 + 2x_1x_2 - 2x_1y_2 - x_2y_2 + x_3y_2$ is minimized. As our last step, we add the polynomials P_1 and P_2 to receive the polynomial $P_3 = P_1 + P_2$. By minimizing P_3, we thus find an assignment of Boolean values to the variables $x_1, x_2, x_3, x_3, y_1, y_2$ that satisfies the formula ϕ. Thus, by arranging the coefficients of P_3 in a matrix, we receive a QUBO matrix representation that, when minimized, yields optimal assignments for ϕ.

2.4 Population-Based Optimization

Let $\mathcal{X}$ be any search space. Let $\mathcal{T}$ be a target space with total order $\leq$. Let $\tau : \mathcal{X} \to \mathcal{T}$ be a target function. A tuple
$\mathcal{E} = (\mathcal{X}, \mathcal{T}, \tau, E, \langle X_t \rangle_{0 \leq t \leq g})$ is a population-based optimization process iff $X_t \subseteq \mathcal{X}$ for all t and E is a possibly randomized or non-deterministic function so that the population-based optimization run is produced by calling E repeatedly, i.e., $X_{t+1} = E(\langle X_u \rangle_{0 \leq u \leq t}, \tau)$ where X_0 is given externally or chosen randomly. [14]

Let $\mathcal{E} = (\mathcal{X}, \mathcal{T}, \tau, E, \langle X_u \rangle_{0 \leq u \leq t})$ be a population-based optimization process. One realization of population-based optimization is the concept of an *Evolutionary Algorithm* (EA). The process $\mathcal{E}$ continues via such an evolutionary algorithm if E has the form

$$E(\langle X_u \rangle_{0 \leq u \leq t}, \tau) = X_{t+1} = selection\big(X_t \cup variation(X_t)\big)$$

where *selection* and *variation* are possibly randomized or possibly non-deterministic functions so that $|selection(X)| \leq |X|$ and $|variation(X)| \geq |X|$ and $\left|selection(X \cup variation(X))\right| = |X|$. In the context of evolutionary search, the term *generation* describes one such iteration of $X_t \rightarrow X_{t+1}$ and *fitness* refers to the evaluation of an individual on the target-function, i.e., $fitness(x) := \tau(x)$. The detailed expression of these operators will follow in Sec. 4.

3 RELATED WORK

To express MAX-3SAT problems as instances of QUBO, researchers have developed many different methods over the past decades. These methods include procedures that are based on polynomial-time reduction of 3SAT problems to instances of maximum independent set [9], procedures that count satisfied clauses [32], and many more [7, 32, 40, 45]. The most common method of transforming MAX-3SAT instances to instances of QUBO is to transform each clause of the MAX-3SAT instance to an instance of QUBO and then sum up all the resulting QUBO matrices, as demonstrated in Sec. 2.2. These approaches include methods representing a given clause as a pseudo-Boolean function and using auxiliary variables to reduce any higher-dimensional (i.e., non-quadratic) terms to quadratic terms. This can be achieved by quadratic reformulation techniques [4, 36]. Furthermore, many approaches, like Chancellor's [7] and Nüßlein's [32] approach, transform each clause to an instance of QUBO according to some custom logic that leads to QUBO instances in which the following **energy condition** holds:

1. If an assignment of Boolean values to the variables of the 3SAT clause satisfies this clause, then this assignment should have minimal energy in this QUBO.
2. If an assignment of Boolean values to the variables of the 3SAT clause does not satisfy this clause, then it must have a higher, non-optimal energy in this QUBO.

This approach introduces one auxiliary variable for each clause of the MAX-3SAT instance. All of these methods have in common that a scientist manually crafted these QUBO transformations.

The Pattern QUBO method [45] is an algorithmic procedure that can identify QUBO representations of 3SAT clauses automatically (i.e., not manually). This procedure can be seen as a generalization of all approaches that transform clauses to instances of QUBO according to the just described logic. As in the case of Chancellor's and Nüßlein's transformation, the Pattern QUBO method introduces an additional auxiliary variable for each clause of the MAX-3SAT instance. To transform an arbitrary clause of a MAX-3SAT problem into an instance of QUBO, the Pattern QUBO method thus starts with an empty 4×4-dimensional QUBO matrix (three variables corresponding to the variables of the 3SAT clause + one auxiliary variable). As QUBO matrices are commonly upper triangular matrices, a 4×4-dimensional QUBO matrix possesses 10 entries. The user next specifies a set of values (e.g., $\{-1, 0, 1\}$) that the Pattern QUBO procedure is allowed to insert into the QUBO matrix. Finally, an exhaustive search procedure tries all possible combinations of assigning values from the specified set of allowed values to the 10 entries of the QUBO matrix to find QUBO matrices that satisfy the above-mentioned *energy condition*. Executing this procedure leads to multiple possibilities of transforming a given 3SAT clause to an instance of QUBO. Choosing any of those possibilities for a given clause results in a mapping from a 3SAT clause to an instance of QUBO. Combining all the QUBO instances (as demonstrated in Sec 2.3) that result from transforming 3SAT clauses to instances of QUBO leads to a QUBO representation of the whole MAX-3SAT problem.

In contrast to the methods explained above, the genetic algorithm method we propose in this paper is a method that can automatically create QUBO representations that fulfill certain criteria (i.e., a predefined sparsity, a given energy gap between incorrect answers and correct solutions) for arbitrary 3SAT clauses. In contrast to the Pattern QUBO method, our genetic algorithm enables finding QUBO representations in much larger spaces, as the Pattern QUBO method uses an exhaustive search procedure that becomes increasingly infeasible the larger the set of possible values specified by the user becomes.

Furthermore, various forms of evolutionary optimization, like genetic algorithms (GA) [17], evolutionary algorithms (EA) [6], memetic algorithms (MA) [30, 31], and other forms of metaheuristic algorithms [41] have recently found application in quantum optimization problems: Both GAs and EAs have been used as classical optimizers for approximate quantum optimization [2, 3] and (random) Ising systems [27, 34]. Other studies utilized evolutionary strategies for the training of quantum classifiers, emphasizing the adaptability of EAs and GAs to quantum machine learning [8, 12]. Noisy quantum effects themselves have even been leveraged in the development of genetic algorithms, e.g., using quantum fluctuation for mutations of individuals [13, 22].

However, even as many of these approaches have leveraged quantum computing (and quantum annealing using the QUBO models) to optimize the solutions and formulation of QUBO models, literature on actual *generation* of QUBO matrices is rather sparse. Quite recently, the AutoQUBO project has proposed ideas for constructing general-purpose optimization QUBOs algorithmically [29, 33], many of which share the concept of decomposing the optimization problem at hand, similar to the Pattern QUBO approach for 3SAT. Similarly, some of the metaheuristic approaches described in the survey of [41] formulate ways to optimize (or decompose) the mathematical QUBO formulations for algorithmic composition of QUBOs, although with more focus on the (assignment-)solution quality rather than the purpose of creating principled QUBO structures like we do in this work.

4 METHOD

We will now detail the approach and parameterization of the population-based evolutionary search that we employ to firstly search correct, principled QUBO patterns and then secondly utilize the found pattern set(s) within an evolutionary selection of patterns for individual clauses in a 3SAT formula. Due to the exploratory nature of this work, we keep the evolutionary operators quite elementary to focus on the validity of the EA search approach in general rather than the fine-tuning of the process.

Our evolutionary buildings blocks *mutate, recombine, select* and *migrate*, serving the purpose of *selection* and *variation* as formalized in Sec. 2.4, are realized as follows for the entirety of our work:

Let *one-point mutate* : $\mathcal{X} \rightarrow \mathcal{X}$ be a randomized mutation function of a single individual (solution candidate) x in the population X.

We write $mutate[\cdot]$ for the per-individual application of *one-point mutate*$(\cdot)$, taking place with probability *mut-rate* $\in [0;1)$. Should mutation occur, one point $i \in [0;|x|)$ from an individual's genome is sampled and the corresponding value-entry is replaced with a valid value-entry of the solution space $\mathcal{X}$.

Let *one-point crossover* $: \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ be a randomized, two-parent recombination function for two parental individuals $x_1, x_2 \sim X$ that produce an offspring $x_3 \in \mathcal{X}$. The new individual combines the first j parts of the solution candidate x_1, i.e, $x_{1,0\ldots j}$, and the remaining part of the solution candidate x_2, i.e., $x_{2,j+1\ldots|x_i|}$, where $j \sim [0;|x_i|)$ is randomly drawn for each parent pair, i.e., $x_3 = x_{1,0\ldots j} +\!\!+ x_{2,j+1\ldots|x_2|}$. Recombined samples are drawn until the population has increased by a percentage of *par-rate* $\in [0;1)$, which we fix to *par-rate*=0.3 for this work. We write *recombine*$[\cdot]$ for the recursive application of *one-point crossover*, for two parental individuals $x_1, x_2 \sim X$ that each get randomly sampled (without replacement) from X and chosen with probability *rec-rate* $\in [0;1)$.

Let $\sigma_N^{roulette} : \wp(\mathcal{X}) \to \wp(\mathcal{X})$ be a randomized selection function that returns $N \in \mathbb{N}$ individuals, i.e., $|\sigma_N(X)| = N$ and $\sigma_N(X) \subseteq X$ for all X. In our case we employ roulette selection, a commonly used selection algorithm (cf. [25, 42, 43]) that selects individuals with a probability proportional to their fitness, i.e., individuals $x \sim X$ are drawn from the population (with replacement) and individual x_i is selected with a probability of $P(x_i) = \frac{fitness(x_i)}{\sum_{x \in X} fitness(x)}$. Upon selection, individual x_i is removed from the population and the selection continues with $\sigma^{roulette}(\{X \setminus x_i\})$. We also employ *elitism*, where we directly include the best percent of the population (*elt-rate* $\in [0;1)$, here *elt-rate* $= 0.01$); therefore at the end of each iteration we select $|\sigma^{roulette}_{|X|-|X|*elt\text{-}rate}(X)| = |X| - |X| * elt\text{-}rate$ non-elite individuals to remain in the population for the next generation.

4.1 Evolutionary Pattern QUBO Creation

As explained in Sec. 3, a 4×4-dimensional QUBO matrix is needed to express a 3SAT clause as an instance of QUBO. Hence a solution candidate Q_C is an upper triangular 4×4 QUBO matrix. For our genetic algorithm, we will represent a solution candidate Q_C as an array $[q_1, q_2, ..., q_{10}]$. Here the values $q_1, ..., q_4$ denote the first row of the upper triangular matrix of Q_C, the values $q_5, ..., q_7$ denote the second row of Q_C, etc. Each array entry can be filled with an arbitrary value from a user-specified predefined value range (e.g., the set $\{-1, 0, 1\}$). Let Q_C be a solution candidate for a type i, $0 \leq i \leq 3$, clause. Let S_{SAT} be the set of all satisfying assignments for a clause of type i. Let E_{SAT} be the set of all energies of all the assignments that satisfy a clause of type i. Let E_{UNSAT} be the energy of the assignment that does not satisfy the clause of type i. Finally let *Energy*(*assignment*) be the energy of an assignment, calculated as described in Sec. 2.2.

We compute the fitness of a solution candidate Q_C via the following fitness functions:

- **Uniformity:** All satisfying assignments of a clause of type i should have the same energy. Let $min\_correct := min\ E_{SAT}$. Penalize deviations with $-\sum_{a \in S_{SAT}} |min\_correct - Energy(a)|$.
- **Correctness:** All satisfying assignments of a clause of type i should have a lower (i.e., better) energy than the non-satisfying assignment. Let S be the set of all correct assignments for a clause of type i with a higher (i.e., worse) energy than the non-satisfying assignment for a clause of type i. Let C be a positive constant. Penalize deviation from the desired outcome with $-\sum_{a \in S} |E_{UNSAT} - Energy(a)| * C + C$.
- **Sparsity:** We want to enforce a specific sparsity sp. Penalize deviations with $-|sp - \sum_{i=0}^{10} \mathbb{I}[x_i \neq 0]|$, where $\mathbb{I}$ is the indicator function of non-zero QUBO matrix entries.
- **Energetic Gap:** We want to enforce a certain gap between the energy of the correct solutions and the energy of the non-satisfying assignment. Let $max\_correct := max\ E_{SAT}$. Let *desired_gap* be a positive integer that represents the desired gap between the worst correct energy and the energy of the non-satisfying assignment. Penalize deviations with $-||max\_correct - E_{UNSAT}| - desired\_gap|$

Our target function $\tau, \mathcal{T} \subseteq \mathbb{R}$, for our EA is then computed by accumulating penalties for combinations of the above fitness criteria. Individuals satisfying all criteria thus have a fitness of 0.

4.2 Evolutionary Pattern QUBO Selection

As explained in Sec. 2.3, to create a QUBO representation of a MAX-3SAT instance, one transforms each clause of a 3SAT formula associated with the MAX-3SAT problem to an instance of QUBO and combines all the resulting QUBOs into one single QUBO instance. Currently known approaches propose a fixed method of transforming each clause **type** to an instance of QUBO. Each of these (four) mappings is then applied to the clauses of the 3SAT formula of the correct type. In this section, we will introduce the idea of mapping each *clause* (not clause type!) individually to a specific instance of a QUBO pattern and then combine the resulting QUBO selections to receive a transformation from a MAX-3SAT problem to an instance of QUBO. We will use an evolutionary method to choose the individual QUBO mapping for each clause of the 3SAT formula as follows. Let m be the number of clauses of the 3SAT instance. First, we are given four ordered sets of valid Pattern QUBOs S_i, $i \in \{0, 1, 2, 3\}$ — one set for each clause type. An individual x of the population X is represented by a list of m integers $x := [l_1, l_2, ..., l_m]$. Each of the integers l_k, $1 \leq k \leq m$, corresponds to an index of the correct Pattern QUBO set. Thus, if the kth clause is of type t_k ($0 \leq t_k \leq 3$), then the kth entry of x (which is l_k) denotes that, for the kth clause, the Pattern QUBO l_k of the set S_{t_k} should be used.

Upon evaluation, the m Pattern QUBOs specified by the individual x are combined into a single QUBO instance (as described in Sec. 2.3) and solved N times using D-Wave's *Advantage_system6.4* quantum annealer [19] directly and D-Wave's tabu search algorithm [20]. Either method returns an assignment $\vec{x} = (x_1 := v_0, \ldots, x_n := v_n), v_i \in \{0, 1\}$, of Boolean values to the variables $x_1, \ldots, x_n$ of the 3SAT formula that the current population is optimizing for. As the target function $\tau, \mathcal{T} \in \mathbb{R}$, — and therefore the fitness of the individual — we simply use the highest count of satisfied clauses across the N assignment trials.

4.3 Baselines

In Sec. 5 we will perform a case study, in which we solve a set of 3SAT formulas with different QUBO formulations on D-Wave's

quantum annealer Advantage_System6.4 and D-Wave's implementation of a tabu search method. The QUBO methods we will compare our result to are:

- Chancellor's method, as described in the special case section of [7].
- Nüßlein's $n + m$ method, as described in [32]
- Random-Fixed-Pattern: Given a set of correct Pattern QUBOs for each of the four types of clauses (see Sec. 2.3), we choose a single Pattern QUBO for each of the four clause types at random and reuse these chosen Pattern QUBOs for all clauses of the respective types within a given 3SAT formula.
- Random-Individual-Pattern: Given a set of correct Pattern QUBOs for each of the four types of clauses. For each individual clause of a given 3SAT formula, we choose an arbitrary Pattern QUBO of the correct type at random.

Our final baseline is directly guessing random solutions for the given 3SAT formulas, without any transformations to QUBO. We will call this baseline simply *random*.

5 EXPERIMENTS

We now layout our experiments in two parts. We first apply the concepts of creating, i.e., searching, valid Pattern QUBOs via the evolutionary approach we have described in Sec. 4.1. With a subset of the found Pattern QUBOs, we then proceed to select individual Pattern QUBOs for each 3SAT clause of a 3SAT formula according to Sec. 4.2 and evaluate their performance in comparison to an ensemble of random baselines, as well as state-of-the-art QUBO transformations for the MAX-3SAT problem. The 'search' and 'selection' aspects correspond to the proposed contributions 1 and 2 respectively.

5.1 Evaluation of EA QUBO Search

We use the EA to search for correct[1] Pattern QUBOs (with the criteria *Uniformity*, *Correctness*) as well as Pattern QUBOs that fulfill specific design criteria, like a specific energy gap (with the *Energetic Gap* criteria, here gap=1), a predefined sparsity (with the *Sparsity* criteria, here sparsity=7), as well as both of them together (with all four criteria), fulfilling both the gap and sparsity requirement. Fig. 1a shows the results of the (integer-range) search spaces $[-1; 1]$, $[-2; 2]$ and $[-3; 3]$. In contrast to satisfying the gap requirement, reaching valid solutions with the correct sparsity is more difficult (esp. for early-generations population), which may also be expected since there are simply fewer valid solutions under this constraint.

We observe that the amount of Pattern QUBOs found decreases as the search space broadens, which is to expected.[2] Evidently, the EA effectively and efficiently identifies more valid patterns in more constrained, smaller search spaces within approximately 1000–2000 total generations. However, it struggles with larger spaces and additional constraints, highlighting the challenges in balancing search space breadth with constraint satisfaction.

Furthermore, Fig. 1b shows that the found Pattern QUBOs are distributed fairly evenly between the four types of 3SAT clauses. For larger ranges, this distribution becomes heavily skewed towards the type 2 and type 3 Pattern QUBOs. Interestingly, as shown in Fig. 1d, this skewing effect is even more pronounced as we drastically increase the search space size to the ranges $[-10; 10]$, $[-20; 20]$, and $[-40; 40]$. This observation is somewhat novel as the initial formulation of the QUBO Types in [45] made no assumptions about their distribution in larger ranges. However, all found correct and uniform Pattern QUBOs of one type are equally valid for our purpose, which is why the question as to why the EA search seems to find such patterns more easily would require a larger-scale ablation study, which we leave for future work.

For the larger search spaces $[-10; 10]$, $[-20; 20]$, and $[-40; 40]$, searching for valid individuals becomes drastically more difficult, which is why we abandon the 'fail fast' idea and give the populations enough generations to converge to good solutions (i.e., correct patterns). Fig. 1c gives an impression of the computational scalability of the evolutionary approach. Larger spaces still remain challenging, but the fact that patterns are found still makes for a promising alternative to brute-force search, which becomes increasingly infeasible as the search space grows.

5.2 Evaluation EA QUBO Selection

We then task the EA to optimize a selection of individual patterns for each clause in a formula from a predefined set per clause-type. Since all 27 valid patterns for range $[-1, 1]$ are known (comparatively quickly found by brute-force search), and we also have already found all of them in the previous experiment (Fig 1b), we provide the EA with the following sets of available patterns to choose from: {type 0: 6, type 1: 7, type 2: 6, type 3: 8} (27 in total).

We construct two datasets to evaluate our approach on. The first one features 100 problems of 500 clauses each and 145 variables (for the commonly used 'difficult' clause-variable ratio of 4.2 [15], where the clauses are sampled uniformly). For the second dataset we construct a set of non-uniformly sampled, empirically difficult 50 formulae with 1000 clauses each and 298 variables according to the *Balanced SAT* method of Spence [37].

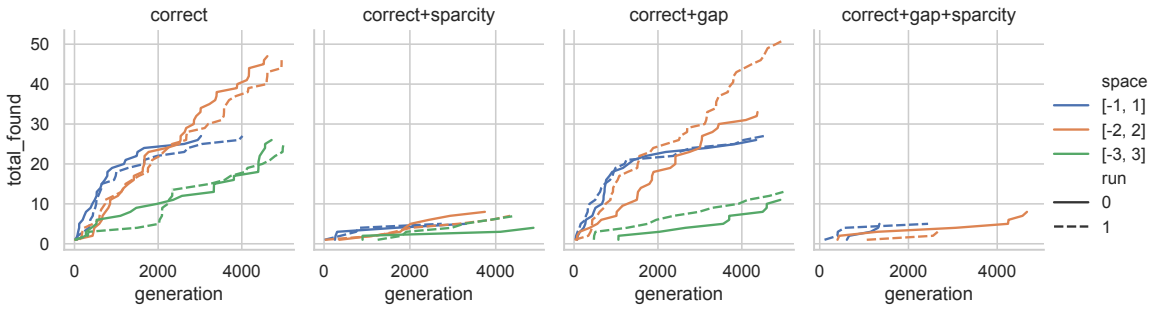
We evaluate the EA (cf. Sec. 4.2) on the 500-dataset against the baselines *chancellor*, *random-individual-pattern*, and *random-fixed-pattern* (cf. Subsec. 4.3). We find that our approach is able to find fully satisfying solutions (500/500 satisfied clauses) much more reliably across the 100 formulae, in comparison to the baselines, where most of the solutions satisfy 2 fewer clauses. Fig. 2a shows the boxen plots[3] of this distribution, where our approach is able to find satisfying solutions for 34 out of 100 3SAT formulae, compared to the baselines which only find 2 satisfying solutions each out of a 100 3SAT formulae.

Similarly, we evaluate the EA classically with tabu-search (5-tabu-reads with 150ms timeout each) on the 1000-dataset against the same baselines *chancellor*, *random-individual-pattern*, and *random-fixed-pattern*, with the same parameters. Results can be seen in Fig. 2b, and although the advantage is not quite as pronounced, the majority of answers is better than or similar in performance to *chancellor*; we observe potential for a few very good solutions

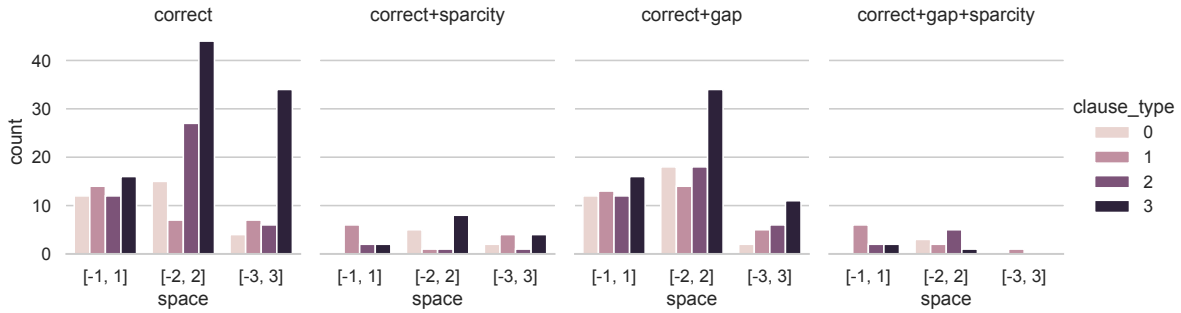
[1] We refer to the criteria combination of *Uniformity* + *Correctness* as 'correct' and 'valid' interchangeably, since *Uniformity* is a generally desired property.

[2] For an intuition, in the range $[-1; 1]$ there are 27 out of 3^{10} valid pattern-value combinations for the four clause types. While it is hard to efficiently determine exactly *how* many correct combinations there are, the space increases drastically upon each range-increase, i.e., $[-2; 2] : 5^{10}$ possibilities, $[-3; 3] : 7^{10}$ possibilities, etc.

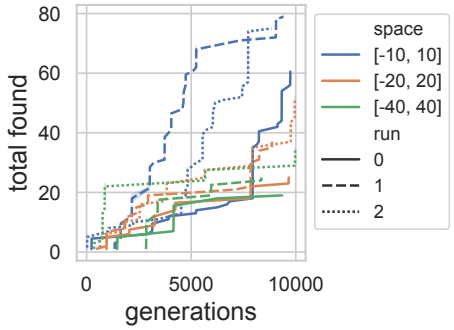
[3] See https://vita.had.co.nz/papers/letter-value-plot.html.



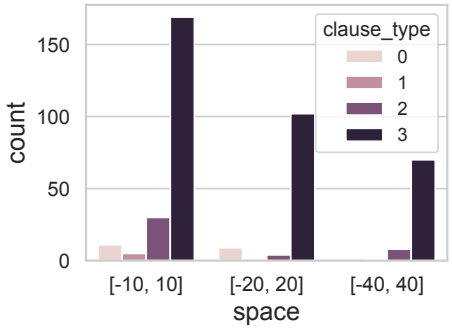
(a) Pattern search over generations (small ranges)



(b) Count of found patterns (small ranges)



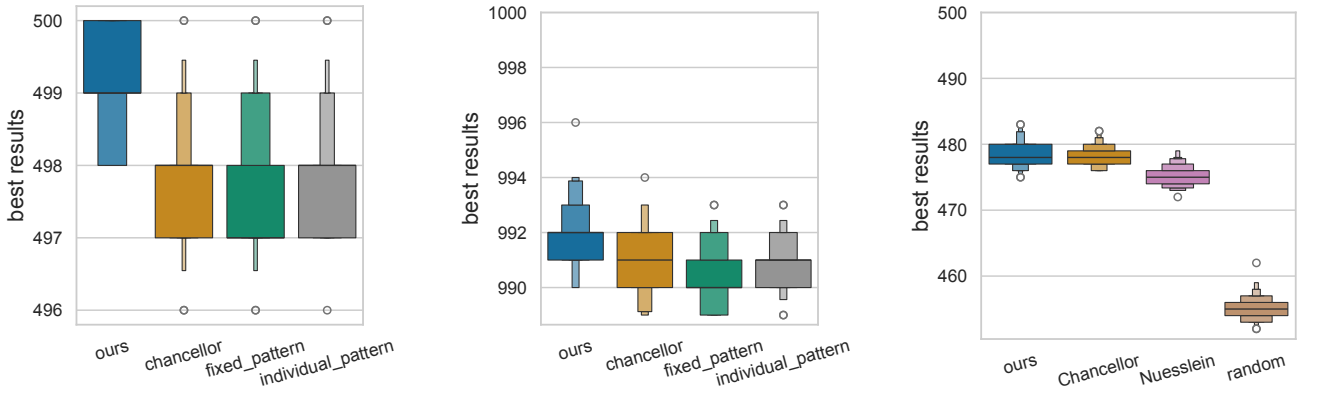
(c) Pattern search over generations (big ranges)



(d) Count of found patterns (big ranges)

Figure 1: Frequency over time (a) and distribution of found pattern types (b) in evolutionary search across three smaller integer spaces: $[-1, 1]$, $[-2, 2]$, and $[-3, 3]$ over 2 runs each. Comparisons are made between searches for correct patterns (b, left), patterns with an energy gap requirement (here gap:1, b, mid-left), patterns with a sparsity requirement (here sparsity:7, b, mid-right), and patterns satisfying both gap and sparsity constraints (b, right). Each run consists of 5000 total generations (500 repetitions of fresh populations evolving over 10 generations, 100 individuals per population with *mut-rate, rec-rate*= 0.5, *elt-rate, mig-rate*= 0.1). To show adaptability to larger ranges we repeat the experiment of finding correct patterns over time (c) and the corresponding count per pattern type (d) with three big spaces ($[-10, 10]$, $[-20, 20]$, and $[-40, 40]$) as well. Each of the 3 runs in (c,d) consists of $10,000$ total generations (100 repetitions of fresh populations evolving over 100 generations, 300 individuals per population with *mut-rate, rec-rate*= 0.5, *elt-rate, mig-rate*= 0.1).

(up to 996/1000 solved clauses). Finally, we evaluate the 500-clause dataset on real quantum hardware (*D-Wave's Advantage_system6.4*, with 10×100 shots), against the baselines *Chancellor*, *Nüsslein* and *random* guessing. Conducting these evaluations used approx. 12 minutes of total QPU time. We find slightly advantageous performance in comparison to *Chancellor*, which is interesting and hints at the transferability of training evolutionary approaches on proxy-solvers like the tabu-search. We also directly outperform the state-of-the-art-approach of *Nüsslein* as well as the *random* baseline.



(a) 500-clause DWAVE tabu-search **(b) 1000-clause DWAVE tabu-search** **(c) 500-clause DWAVE *Advantage_system6.4***

Figure 2: Evolutionary pattern selection across datasets with varying clause length and clause complexity. (a) Performance distribution on a 500-clause dataset with a standard difficulty ratio, showcasing the number of clauses satisfied by our EA (blue) in comparison to baseline methods *chancellor* (gold), *fixed-patterns* (teal) and *individual-pattern* (grey). (b) Outcomes on a more complex 1000-clause dataset against the same baselines. (c) Performance of the EA on a 500-clause dataset using the D-Wave Advantage_system6.4 quantum computer, compared to *chancellor*, *nüsslein* (violet) and *random* guessing (brown) indicating the transfer-ability of EA optimizations from proxy solvers to actual quantum hardware. Results are obtained using parameters of 100 individuals over 50 generations, 5-tabu reads with 150ms timeout for the EA, and best results of $50 * 5$-tabu reads (also 150ms timeout) for all other baselines.

6 CONCLUSION & FUTURE WORK

In this paper we proposed two methods of using evolutionary algorithms to create QUBO representations of MAX-3SAT problems automatically. As our first method, we proposed an evolutionary algorithm for creating Pattern QUBOs for any 3SAT clause in given value ranges. Especially for the commonly used smaller ranges we can quickly and efficiently find valid patterns. Furthermore, we showed how to adapt the fitness function of this method such that the method specifically creates Pattern QUBOs that fulfill certain design criteria, like a user-specified sparsity or a specific energy gap between correct and incorrect assignments. Then, assuming a set of Pattern QUBOs for each type of 3SAT clause, our second evolutionary algorithm can create full QUBO representations of MAX-3SAT problems, by selectively choosing a Pattern QUBO from the given set of Pattern QUBOs, one for each clause of the 3SAT problem individually. This is in contrast to currently known procedures of transforming MAX-3SAT instances to instances of QUBO, where each type of clause (i.e., zero negations, one negation,...) of the MAX-3SAT problem is transformed to an instance of QUBO by using a single, predefined Pattern QUBO. This method of assembling QUBOs outperforms our baselines for smaller 500-clause datasets, draws or improves on the performance for longer, more difficult 1000-clause problems, and even shows competitive performance to state-of-the-art baselines when evaluated on real quantum hardware, highlighting potential for transferring classically optimized (tabu-search) populations to noisy quantum optimizers.

In the future, we aim to delve into optimizing evolutionary strategy parameters, such as population sizes, mutation rates, and different fitness functions to enhance the QUBO generation process for MAX-3SAT problems. This optimization may also include more advanced and efficient evolutionary operators, rather than the simple roulette-selection and one-point crossover that we have employed for the proof-of-concept in this paper. We will also investigate why our evolutionary method generates significantly more Pattern QUBOs for type 3 clauses than for any other type of clause.

In this paper, we trained the second evolutionary algorithm via the classical tabu search method. In the future, we want to train our second evolutionary method directly on D-Wave's quantum annealer to guide the evolutionary search towards creating QUBO representations of MAX-3SAT problems that perform well when solved on the quantum annealer. Finally, by choosing Pattern QUBOs for each individual clause of a 3SAT formula, it may be possible to create QUBO representations of MAX-3SAT problems that possess a certain sparsity or a (more) uniform value range (i.e., the maximum and minimum value of the QUBO entries are not too far apart). We will evaluate adding these criteria as an extension to the fitness function of our second evolutionary method.

7 ACKNOWLEDGMENTS

The partial funding of this paper by the German Federal Ministry of Education and Research through the funding program "quantum technologies — from basic research to market" (contract number: 13N16196) is gratefully acknowledged. Furthermore, this paper was also partially funded by the German Federal Ministry for Economic Affairs and Climate Action through the funding program "Quantum Computing – Applications for the industry" (contract number: 01MQ22008A).

REFERENCES

[1] Pietro Abate, Roberto Di Cosmo, Georgios Gousios, and Stefano Zacchiroli. 2020. Dependency solving is still hard, but we are getting better at it. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 547–551.

[2] Giovanni Acampora, Angela Chiatto, and Autilia Vitiello. 2023. A Comparison of Evolutionary Algorithms for Training Variational Quantum Classifiers. In *2023 IEEE Congress on Evolutionary Computation (CEC)*. 1–8. https://doi.org/10.1109/CEC53210.2023.10254076

[3] Giovanni Acampora, Angela Chiatto, and Autilia Vitiello. 2023. Genetic algorithms as classical optimizer for the Quantum Approximate Optimization Algorithm. *Applied Soft Computing* 142 (2023), 110296.

[4] Martin Anthony, Endre Boros, Yves Crama, and Aritanan Gruber. 2017. Quadratic reformulations of nonlinear binary optimization problems. *Mathematical Programming* 162 (2017), 115–144.

[5] Sanjeev Arora and Boaz Barak. 2009. *Computational complexity: a modern approach*. Cambrtidge University Press.

[6] Thomas Bäck and Hans-Paul Schwefel. 1993. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation* 1, 1 (1993), 1–23.

[7] Nicholas Chancellor, Stefan Zohren, Paul A Warburton, Simon C Benjamin, and Stephen Roberts. 2016. A direct mapping of Max k-SAT and high order parity checks to a chimera graph. *Scientific reports* 6, 1 (2016), 37107.

[8] Samuel Yen-Chi Chen, Chih-Min Huang, Chia-Wei Hsing, Hsi-Sheng Goan, and Ying-Jer Kao. 2022. Variational quantum reinforcement learning via evolutionary optimization. *Machine Learning: Science and Technology* 3, 1 (Feb. 2022), 015025. https://doi.org/10.1088/2632-2153/ac4559

[9] Vicky Choi. 2010. Adiabatic quantum algorithms for the NP-complete maximum-weight independent set, exact cover and 3SAT problems. *arXiv preprint arXiv:1004.2226* (2010).

[10] Stephen A Cook. 1971. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*. 151–158.

[11] Sebastian Feld, Christoph Roch, Thomas Gabor, Christian Seidel, Florian Neukart, Isabella Galter, Wolfgang Mauerer, and Claudia Linnhoff-Popien. 2019. A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer. *Frontiers in ICT* 6 (2019), 13.

[12] Lucas Friedrich and Jonas Maziero. 2023. Learning to learn with an evolutionary strategy applied to variational quantum algorithms. arXiv:2310.17402 (Oct. 2023). https://doi.org/10.48550/arXiv.2310.17402 arXiv:2310.17402 [quant-ph].

[13] Thomas Gabor, Michael Lachner, Nico Kraus, Christoph Roch, Jonas Stein, Daniel Ratke, and Claudia Linnhoff-Popien. 2022. Modifying the quantum-assisted genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2205–2213.

[14] Thomas Gabor, Thomy Phan, and Claudia Linnhoff-Popien. 2021. Productive fitness in diversity-aware evolutionary algorithms. *Natural Computing* 20, 3 (2021), 363–376.

[15] Ian P Gent and Toby Walsh. 1994. The SAT phase transition. In *ECAI*, Vol. 94. PITMAN, 105–109.

[16] Fred Glover, Gary Kochenberger, and Yu Du. 2018. A tutorial on formulating and using QUBO models. *arXiv preprint arXiv:1811.11538* (2018).

[17] John H. Holland. 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press. https://books.google.de/books?hl=de&lr=&id=5EgGaBkwvWcC&oi=fnd&pg=PR7&dq=%5D+J.H.+Holland,+Adaptation+in+Natural+and+Artificial+Systems&ots=mKgk2ZLqxn&sig=JGTmkvUivw5PRIgaQwpzfXYQHkY

[18] Kazuki Ikeda, Yuma Nakamura, and Travis S Humble. 2019. Application of quantum annealing to nurse scheduling problem. *Scientific reports* 9, 1 (2019), 12837.

[19] D-Wave Systems Inc. 2024. *DWAVE Quantum Annealing*. https://docs.dwavesys.com/docs/latest/c_gs_2.html

[20] D-Wave Systems Inc. 2024. *DWAVE Tabu-Search*. https://docs.ocean.dwavesys.com/projects/tabu/en/latest/

[21] Hirotaka Irie, Goragot Wongpaisarnsin, Masayoshi Terabe, Akira Miki, and Shinichirou Taguchi. 2019. Quantum annealing of vehicle routing problem with time, state and capacity. In *Quantum Technology and Optimization Problems: First International Workshop, QTOP 2019, Munich, Germany, March 18, 2019, Proceedings 1*. Springer, 145–156.

[22] James King, Masoud Mohseni, William Bernoudy, Alexandre Fréchette, Hossein Sadeghi, Sergei V. Isakov, Hartmut Neven, and Mohammad H. Amin. 2019. Quantum-Assisted Genetic Algorithm. arXiv:1907.00707 (June 2019). http://arxiv.org/abs/1907.00707 arXiv:1907.00707 [quant-ph].

[23] Tom Krüger and Wolfgang Mauerer. 2020. Quantum annealing-based software components: An experimental case study with sat solving. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 445–450.

[24] Jonas Lang, Sebastian Zielinski, and Sebastian Feld. 2022. Strategic portfolio optimization using simulated, digital, and quantum annealing. *Applied Sciences* 12, 23 (2022), 12288.

[25] Adam Lipowski and Dorota Lipowska. 2012. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications* 391, 6 (2012), 2193–2196.

[26] Andrew Lucas. 2014. Ising formulations of many NP problems. *Frontiers in physics* 2 (2014), 74887.

[27] A. Z. Maksymowicz, J. E. Galletly, M. S. Magdon, and I. L. Maksymowicz. 1994. Genetic algorithm approach for Ising model. *Journal of magnetism and magnetic materials* 133, 1–3 (1994), 40–41.

[28] João P Marques-Silva and Karem A Sakallah. 2000. Boolean satisfiability in electronic design automation. In *Proceedings of the 37th Annual Design Automation Conference*. 675–680.

[29] Alberto Moraglio, Serban Georgescu, and Przemysław Sadowski. 2022. AutoQubo: data-driven automatic QUBO generation. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, Boston Massachusetts, 2232–2239. https://doi.org/10.1145/3520304.3533965

[30] Pablo Moscato, Carlos Cotta, and Alexandre Mendes. 2004. Memetic algorithms. *New optimization techniques in engineering* 141 (2004), 53–85.

[31] Ferrante Neri and Carlos Cotta. 2012. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation* 2 (2012), 1–14.

[32] Jonas Nüßlein, Sebastian Zielinski, Thomas Gabor, Claudia Linnhoff-Popien, and Sebastian Feld. 2023. Solving (Max) 3-SAT via Quadratic Unconstrained Binary Optimization. *arXiv preprint arXiv:2302.03536* (2023).

[33] Justin Pauckert, Mayowa Ayodele, Marcos Diez García, Serban Georgescu, and Matthieu Parizy. 2023. AutoQUBO v2: Towards Efficient and Effective QUBO Formulations for Ising Machines. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation (GECCO '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 227–230. https://doi.org/10.1145/3583133.3590662

[34] Adam Prügel-Bennett and Jonathan L. Shapiro. 1997. The dynamics of a genetic algorithm for simple random Ising systems. *Physica D: Nonlinear Phenomena* 104, 1 (1997), 75–114.

[35] Jussi Rintanen. 2012. Planning as satisfiability: Heuristics. *Artificial intelligence* 193 (2012), 45–86.

[36] Ivo G Rosenberg. 1975. REDUCTION OF BIVALENT MAXIMIZATION TO THE QUADRATIC CASE. (1975).

[37] Ivor Spence. 2017. Balanced random SAT benchmarks. *SAT COMPETITION 2017* (2017), 53.

[38] Davide Venturelli and Alexei Kondratyev. 2019. Reverse quantum annealing approach to portfolio optimization problems. *Quantum Machine Intelligence* 1, 1 (2019), 17–30.

[39] Davide Venturelli, Dominic JJ Marchand, and Galo Rojo. 2015. Quantum annealing implementation of job-shop scheduling. *arXiv preprint arXiv:1506.08479* (2015).

[40] Amit Verma, Mark Lewis, and Gary Kochenberger. 2021. Efficient QUBO transformation for higher degree pseudo Boolean functions. *arXiv preprint arXiv:2107.11695* (2021).

[41] Yang Wang and Jin-Kao Hao. 2022. Metaheuristic algorithms. In *The Quadratic Unconstrained Binary Optimization Problem: Theory, Algorithms, and Applications*. Springer, 241–259.

[42] Fengrui Yu, Xueliang Fu, Honghui Li, and Gaifang Dong. 2016. Improved roulette wheel selection-based genetic algorithm for TSP. In *2016 International conference on network and information systems for computers (ICNISC)*. IEEE, 151–154.

[43] Liming Zhang, Huiyou Chang, and Ruitian Xu. 2012. Equal-width partitioning roulette wheel selection in genetic algorithm. In *2012 Conference on Technologies and Applications of Artificial Intelligence*. IEEE, 62–67.

[44] Sebastian Zielinski, Jonas Nüßlein, Jonas Stein, Thomas Gabor, Claudia Linnhoff-Popien, and Sebastian Feld. 2023. Influence of Different 3SAT-to-QUBO Transformations on the Solution Quality of Quantum Annealing: A Benchmark Study. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (Lisbon, Portugal) *(GECCO '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 2263–2271. https://doi.org/10.1145/3583133.3596330

[45] Sebastian Zielinski, Jonas Nüßlein, Jonas Stein, Thomas Gabor, Claudia Linnhoff-Popien, and Sebastian Feld. 2023. Pattern QUBOs: Algorithmic Construction of 3SAT-to-QUBO Transformations. *Electronics* 12, 16 (2023). https://doi.org/10.3390/electronics12163492