Quantum Annealing-Based Software Components:
An Experimental Case Study with SAT Solving

Tom Krüger
Ulm University
Germany
tom.krueger@uni-ulm.de

Wolfgang Mauerer
Technical University of Applied Sciences Regensburg
Siemens AG, Corporate Research, Munich
Germany
wolfgang.mauerer@othr.de

ABSTRACT

Quantum computers have the potential of solving problems more efficiently than classical computers. While first commercial prototypes have become available, the performance of such machines in practical application is still subject to exploration. Quantum computers will not entirely replace classical machines, but serve as accelerators for specific problems. This necessitates integrating quantum computational primitives into existing applications.

In this paper, we perform a case study on how to augment existing software with quantum computational primitives for the Boolean satisfiability problem (SAT) implemented using a quantum annealer (QA). We discuss relevant quality measures for quantum components, and show that mathematically equivalent, but structurally different ways of transforming SAT to a QA can lead to substantial differences regarding these qualities. We argue that engineers need to be aware that (and which) details, although they may be less relevant in traditional software engineering, require considerable attention in quantum computing.

KEYWORDS

Quantum Computing, Quantum Annealing, Boolean Satisfiability, Experimental Performance Analysis

1 INTRODUCTION

The upcoming end of Moore's law and the trend towards energy efficient systems, but the likewise ever-growing need for more computational power pose substantial challenges to systems engineering and software architecture. New computational approaches that substantially diverge from technologies established during the last decades start to graduate from research laboratories into first working prototypes. Especially quantum computing has gained substantial attraction during the last years [24]. Programming quantum

computers (QC) differs drastically from previously established techniques and approaches. Integrating QC into existing appliances must not only matter be addressed at the level of algorithmic implementation, but also concerns many of the broader issues investigated in software engineering [4]. In this paper, we argue that the problem at the current stage of development must be considered at a much lower level of abstraction than is customary in software engineering, and illustrate this by a case study of how to transition a core computational primitive—solving binary satisfiability problems—from classical to quantum in existing software architectures. Our study illustrates that defining and testing specific quality properties of QC components is one of the crucial challenges. These properties do not play a central role in traditional engineering, but must be considered in software architectures with quantum components. We illustrate this by analysing different approaches—one of which has been specifically designed for this paper, and improves considerably on the state-of-the-art—to solving the binary satisfiability (SAT) problem. We hope this helps readers to form a realistic intuition of near- and mid-term capabilities, potentials and challenges of augmenting software with quantum components.

2 QUANTUM ANNEALING

By utilising quantum mechanical properties, QCs are expected to solve some problems more efficiently than their classical counterparts. Simulations of quantum systems [23] and chemical reactions [25], breaking of cryptographic codes [28], but also optimising portfolios [37] are among the list of candidate problems. Recent advances—although not undisputed—claim *quantum supremacy* [3], even if for extremely artificial problems. Real-world adoption of quantum computing, as it matters to software engineering, is likely to happen in an evolutionary way than by disruptive revolution.

We base our considerations on quantum *annealers*: Many early potential industrial use-cases [22, 31, 32] rest on this class of machines, in part because they were among the first offerings available for commercial use (discussions about the full quantum mechanical nature of such machines [27] are not relevant for our purposes).

Especially NP-complete problems, which are known to be classically intractable for inputs of growing size when non-approximate solutions are desired, are candidates for which (polynomial) quantum speed-ups would be desirable. Many NP-complete problems of practical interest are known. Especially the Boolean satisfiability problem (SAT) has received substantial attention because many use cases, from system verification to constrained planning problems [13], have SAT at their core. Quantum annealers are particularly well suited to process problems of this type [16]. They

differ considerably from gate-based approaches in their physical realisation, and in the ways programs are engineered.

## 2.1 Using Quantum Annealing Primitives in Existing Software Architectures

Software engineering is (ignoring many aspects that we cannot address for the lack of space) concerned with development, integration, and testing (verification, validation, performance, quality, …) of software [4]. This impacts quantum software development:

2.1.1 Development. A considerable body of previous research devoted to developing languages for programming quantum systems focuses on gate-based approaches (*e.g.*, [11, 30, 34]). A growing number of quantum programming languages has been devised for this hardware class (*e.g.*, [10, 17, 33]). Roughly speaking, gate based quantum computers relate to quantum annealers like imperative programming languages do to declarative approaches. A deep understanding of quantum mechanics is not required to use current QA hardware, which is beneficial from a software point of view. Engineers can resort to techniques known from constraint programming, optimisation, and problem reduction.

2.1.2 Integration. As Knill [14] discussed as early as 1996, quantum computers will not entirely replace traditional machines, but will be part of hybrid quantum-classical architectures, not unlike GPUs [1, 12] or other accelerators (TPUs, FPGAs, …). Quantum annealers can be seen as hardware accelerators for approximating quadratic unconstrained binary optimisation (QUBO) problem.

The ability to easily replace functional components of a software architecture is a crucial element of component-based software engineering [35, 36], and many existing applications are designed along these ideas. In the following, we consider that a SAT solving component is supposed to be replaced by a QA device in an existing software architecture.

Replacing a library function call to a traditional solver by a network-based job submission to a QA device is an easy programming tasks that we do not consider any further. However, two data conversions are necessary, as Figure 1 illustrates: The propositional calculus formula for which a solution is sought must be mapped to a QUBO. Once the result of the optimisation process is available, is must be translated back to the original SAT model. Both steps can be trivially abstracted by an interface.
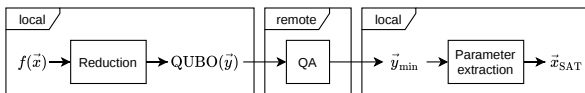
Figure 1: Interface wrapping: Classical SAT solvers can be replaced with a QA based implementation with limited effort.

2.1.3 Testing. Miranskyy and Zhang [20] discuss testing aspects related to verification and validation of quantum programs. Fundamental properties of NP-complete problems guarantee that solutions can be verified in polynomial time [29], and consequently, validation and verification of QA programs is not a core challenge.

However, quantum annealers usually only deliver *approximate* solutions to problems, and the quality of approximation is closely related to how "programs" (in the form of mathematical reductions) are created. We focus on the issues arising from this scenario in the rest of the paper.

2.2 Workflow

The workflow for solving problems on quantum annealers is more involved than for classical constraint optimisation. The necessary process comprises five stages, and choices in some of the stages can greatly influence performance and accuracy of computations. Consequently, some knowledge of the inner working of the quantum annealing process are useful. An AQO computation proceeds along the following stages [12, 18]:

PROBLEM REDUCTION: Like classical constraint optimization solvers, QA machines can optimise a specific class of models. Annealers can find solutions to *quadratic unconstrained binary optimization* (QUBO) problems [15], which are given by

$$\min[\vec{x}]\left(\sum_i c_{ii}x_i + \sum_{i,j} c_{ij}x_ix_j\right) \quad (1)$$

with $x \in \{0, 1\}$ and $c \in \mathbb{R}$. A QUBO can be represented by a weighted graph with nodes x_i and associated weights c_{ii}. Weighted edges are given by c_{ij}.

Reducing a given problem p to a QUBO, $p \leq$ QUBO, requires no knowledge of quantum mechanics, and is similar to well-known reductions to Boolean satisfiability problems. As we will discuss later, structurally different (but logically equivalent) reductions can lead to drastically different performance on contemporary hardware.

HARDWARE EMBEDDING: Software solvers can react dynamically to input, and easily build arbitrary data structures. For QA, the "data structure" used to represent a given input is fixed in hardware. This step "translates" an input onto the hardware structure [1, 7] (see Figure 2). Mathematically equivalent reductions can lead to pronounced differences in solution quality, as we show in Section 3.

HARDWARE PROGRAMMING: The problem embedding needs to be transferred to the machine. The physical details of this operation are irrelevant to programmers, except that some parameters—most importantly, the duration of the annealing process—can be influenced. Finding an optimal value is currently a matter of experimentation.

EXECUTION: The machine finds a solution to the optimisation problem by "executing" a physical process.

POST PROCESSING: Results obtained in the previous step are usually only *close* to the desired optimum. Classical post-processing can improve solution quality [9]. We will ignore this step in this paper since we are interested in the capabilities of QA as such, and not of classical data processing.

2.3 Experimentation Platform

All experiments that we discuss in the following were performed on a D-Wave 2000Q quantum annealer, model DW_2000Q_2_1. The machine can be remotely accessed via a Python-based API. Performing computations requires to specify a problem QUBO, and (essentially) anneal time and desired sample size n. Once the QA
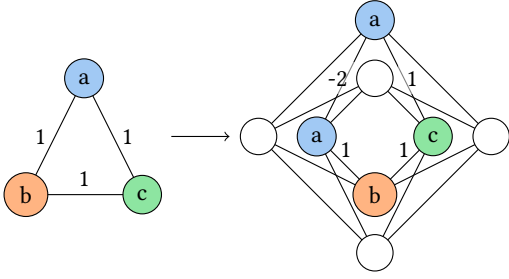
Figure 2: Example for embedding a *logical* graph that describes couplings between qubits (left) into a *physical* qubit structure (right) with limited connectivity. Node "a" is mapped to a chain of two nodes representing "a", which illustrates that the amount of physical qubits required to represent a problem is larger than the amount needed for a structural description.

has evaluated the problem, a result set with n samples is returned. Each sample contains an assignment for all qubits.

While it is possible to arbitrarily weigh the interaction between qubits as specified by term c_{ij} from Eq. (1), there are substantial restrictions on which qubit i can *physically* and *directly* interact with which qubits j (see Ref. [5] for details on the available hardware graph structure). This limited connectivity poses a major practical challenge when mapping logical to physical problems, since a pair of nodes that requires a logical connection (a non-zero entry c_{ij} in Eq. (1)) must be represented by a chain of multiple nodes on the hardware graph. This considerably limits the number of *effectively usable* qubits as compared to the number of *physically available* qubits—Figure 6 exemplifies the problem visually. In general, longer chains lead to more undesirable physical perturbance, and decrease result quality [26]. As a rule of thumb, the number of usable logical qubits is only about 5-10% the number of physical qubits.

3 QUALITY ASSESSMENT OF QUANTUM 3-SAT

Let us now turn our attention to discussing how implementation details of quantum computational primitives can influence qualities of software. We focus on the problem of finding and comparing reductions of the problem to a machine specific structure. Such low-level issues are usually not of much relevance in software engineering, and are justifiably perceived as implementation details—however, we show that this level of abstraction is far from reached on quantum machines yet.

The k-SAT problem, the cornerstone of NP-completeness [13], serves as an example. We first discuss different reductions of k-SAT $\leq_P$ QUBO, and show how differences arise from seemingly small details. We then offer guidance on comparing reductions.

3.1 Problem Definition

The problem of Boolean satisfiability is well known: Let $X := \{x_1, x_2, \ldots, x_n\}$ be a set of Boolean variables, and let *literals* be defined as $L := \{l | l \in \{x, \overline{x}\}, x \in X\}$. The set of all *clauses* is given by $C := \{C_i | i \in [1; n], C_i \subset L, |C_i| = k\}$. For each $x_i \in X$, there exists
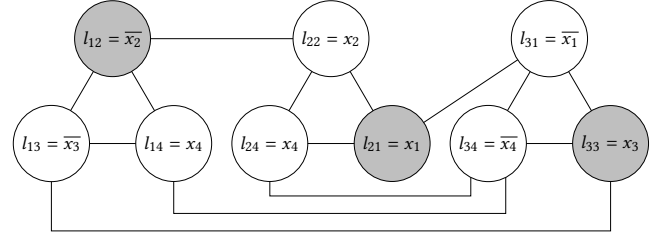

Figure 3: Graphical illustration of a QUBO formula that represents $f(\vec{x}) = (\overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overline{x_1} \vee x_3 \vee \overline{x_4})$ using Choi's reduction. Grey nodes represent a satisfying assignment $[x_1 \mapsto 1, x_2 \mapsto 0, x_3 \mapsto 1]$.

at least one C_j such that $x_i \in C_j$. A function $f(\vec{x}) = \bigwedge_{C_i \in C} \bigvee_{l \in C_i} l$ that satisfies these conditions is called a k-CNF function. Given a k-CNF function $f(\vec{x})$, the k-SAT problem is to find an assignment $\vec{x}_t$ such that $f(\vec{x}_t) = true$. It is textbook knowledge [29] that every CNF formula can be cast in 3-CNF form.

The k-SAT problem is the cornerstone of NP-completeness, but not all specific instances are difficult to solve. The hardness of an instance depends on the ratio of clauses per variable $\alpha = \frac{|C|}{|X|}$ [6]. For instances with few clauses per variable (small α), it is easy to find satisfying assignments. For instances with many clauses per variable, it is easy to find contradictions. Instances with large or small values of α tend to be easy to solve. In an α region surrounding $\alpha_c \approx 4.25$, the probability that a random k-SAT formula can be satisfied drops abruptly from 1 to 0 [6, 21], and the hardest instances are contained in this parameter regime. Improvements in SAT solving are therefore most desirable around this *phase transition*.

3.2 Choi's Standard Reduction

Choi [8] gives a standard reduction from k-SAT to a QUBO. Let l_{ij} be the literal of variable x_j in clause i. Two literals l_{ij} and $l_{i'j}$ are *in conflict* if $l_{ij} = \bar{l}_{i'j}$. Satisfying a Boolean function in CNF implies at least one satisfied, but conflict-free literal per clause.

The reduction assigns a negative weight $-\omega$ to $l_{ij}: -\sum_{l_{ij}} \omega l_{ij}$. All literals of a clause are fully connected with positive weighted edges: $\sum_{l_{ij}, l_{ij'} \in C_i} \delta l_{ij} l_{ij'}$. All conflicting literals of the same variable are pairwise connected with *positive* edge weights: $\sum_{l_{ij}=\bar{l}_{i'j}} \delta l_{ij} l_{i'j}$. The last two sums are pure penalty terms and evaluate to 0 for correct assignments. This leads to a definition illustrated in Figure 3:

Definition 3.1 (k-SAT $\leq_P$ QUBO (MIS)). Let $f(\vec{x})$ be a boolean k-CNF function. The literal of a variable $x_j \in \vec{x}$ in a clause $C_i \in C$ is given by $l_{ij} \in \{0, 1\}$. Under the constraint $\forall \delta, \omega : \delta > \omega > 0$,

$$\min[\vec{x}] \left(-\sum_{l_{ij}} \omega l_{ij} + \sum_{l_{ij}, l_{ij'} \in C_i} \delta l_{ij} l_{ij'} + \sum_{l_{ij}=\bar{l}_{i'j}} \delta l_{ij} l_{i'j} \right) \quad (2)$$

finds a satisfying assignment for $f(\vec{x})$ if one exists.[8]

3.3 Backbone Reduction

To demonstrate the effect of different reductions on various aspects of QA performance, consider a different reduction that we have

devised for this paper, and that improves (as we will analyse later) on the reduction given in Eq. (2):

Definition 3.2 (k-SAT $\leq_P$ QUBO (Backbone)). Let $f(\vec{x})$ be a Boolean function in k-CNF, and let l_{ij} be a literal of $x_j \in \vec{x}$ in $C_i \in C$, with $l_{ij}, x_j \in \{0, 1\}$. Then

$$q(\vec{x}) = \omega \left(\sum_{l_{ij}, l_{ij'}} l_{ij}l_{ij'} + \sum_{l_{ij}=x_j} -l_{ij}x_j + \sum_{l_{ij} \neq x_j} -l_{ij} + l_{ij}x_j \right) \quad (3)$$

with $\omega > 0$ describes a QUBO $q(\vec{x})$ for which $\min[\vec{x}]$ represents a satisfying assignment of $f(\vec{x})$ if one exists.

Mathematical details of the derivation are given in Appendix A.

3.4 Quality Criteria for Reductions

Quality criteria for software are plentiful, and many of them also apply to the relative merits of reductions. Since the development of quantum computers is mainly driven by the desire for more computational power, we focus on two indicators: Performance and scalability. There is (despite recent standardisation efforts) no universally applicable (and accepted) definition of how to measure performance of quantum computers; this is particularly hard for QA, were the run-time is not determined by the input, but chosen as a parameter—the annealing time. Consequently, we consider solution *quality*—how likely is it to obtain a correct answer that does not violate constraints, and how accurate is the answer (*i.e.*, how close is it to the optimal achievable value)—as proxy for performance. Scalability considers the question of how large problems can be solved on a hardware of given size (*i.e.*, number of physical qubits).

The achievable accuracy of a reduction depends on its structure (how well do logical connections between qubits match the available physical structure?) and on hardware parameters. While the adiabatic theorem ensures that longer annealing times (runtimes) results in better accuracy, flaws and approximate implementations of the scheme in real hardware lead to less direct relations. Like with traditional approximation algorithms, increasing the amount of computes samples also leads to more accurate solutions.

## 3.5 Generating Instance Datasets

Owing to the lack of a published, physically accurate model of the quantum annealer that includes imperfections and noise,[1] determining scalability and accuracy is currently only possible with experimental means [19].

When SAT is used to model constrained optimisation problems in practical applications, the resulting SAT instances often exhibit specific structural properties, which can guide the generation of useful test instances for determining quality properties of reductions. This is, of course, not unlike the well-studied problem of generating tailored input data for general software testing problems [2].

We are interested in a general comparison of reductions, and therefore base our input data generation on general properties of 3-SAT. We have discussed that the problem exhibits different regimes in Section 3.1, and systematically generate random 3-SAT instances that cover these by sweeping across different values of α. For the

---

[1] It is unlikely that such a physically accurate model will be available in the near- or mid-term future.
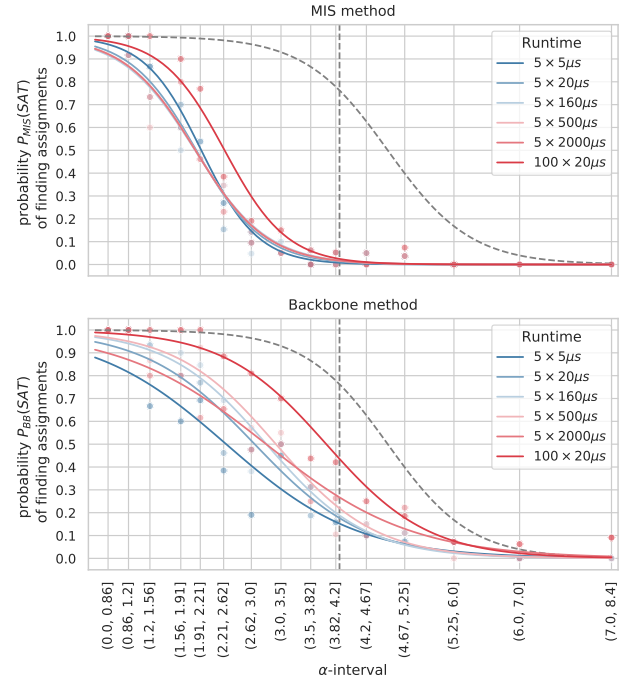
Figure 4: Influence of the embedding method on the probability of finding correct satisfying assignments for randomly generated 3-SAT instances with varying ratios α of clauses to variables. The horizontal dashed lines marks the critical value α_c (accompanied by a peak increase in required computing time when using traditional numeric solvers). The dashed curve represents the probability distribution of finding a satisfying assignment with optimal solvers. To ease comparing quantum and classical result, a logistic regression curve is given for each parameter variation. Both (mathematically equivalent) methods arrive at correct conclusions less often than classical solvers, which is caused by imperfections and limitations of the available hardware.

number of needed qubits, $k|C|$ is the dominant term for both reduction approaches. Keeping $|C|$ fixed and varying $|V|$ produces stable QUBO sizes across the α-spectrum, which guarantees a consistent hardware graph utilization.

3.6 Experimental Results

We generate a data-set containing 250 random 3-SAT instances with 42 clauses each. In total, six runs with varying annealing times (5 to 2000μs) and samples sizes (5 and 100) were performed on the quantum annealer described in Section 2.3. Figure 4 shows results for the two different reduction methods.

3.6.1 Accuracy. For Choi's MIS-based reduction, the annealing time does not substantially effect the accuracy. Using probability amplification by performing a larger number of runs 100×20μs, does improve the accuracy slightly. Results obtained with the backbone method, in contrast, improve with increasing annealing time, and
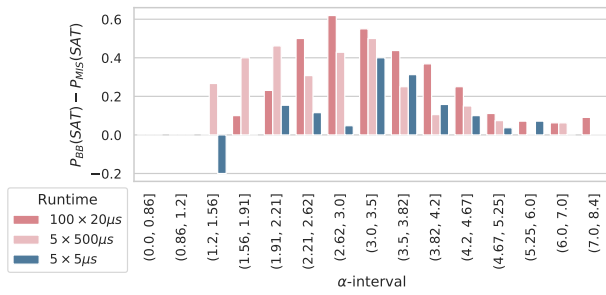
Figure 5: Accuracy difference (in percentage points) between MIS and backbone method.
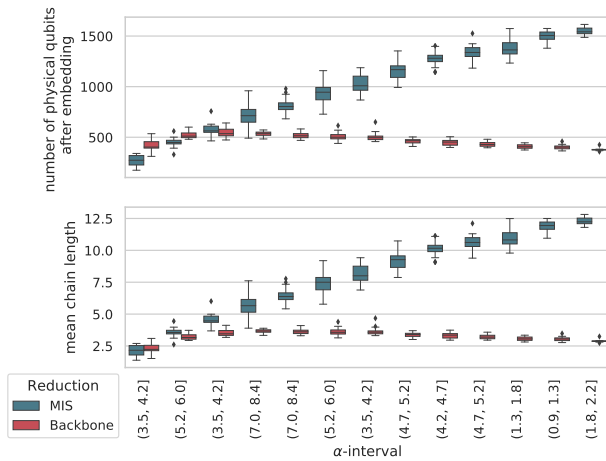

Figure 6: Number of required physical qubits to after embedding a QUBO for a given ratio of variables and clauses (top) and median chain lengths necessary to connect qubits without direct interconnections (bottom).

increasing the number of runs is also accompanied by a larger improvement as compared to the MIS method. It is also important to note that using an excessively long annealing time of $2000\mu s$ results in a *decrease* of result quality[2].

Recall from the above discussion that solving SAT instances in the critical parameter region around α_c is most involved for classical solvers, and improvements by quantum computers are most desirable in this region. Unfortunately, the MIS method delivers satisfying solutions in this range with almost zero probability.

Figure 4 directly compares accuracy results. The difference in accuracy reaches up to 60%, and the backbone method is consistently more accurate for all scenarios. The decreasing difference in accuracy at $\alpha > \alpha_c$ is a consequence of the low number of satisfiable instances in this region. Around the critical region, we observe marked differences of around 35%.

---

[2]This observation does contradict the adiabatic theorem; the effect is likely caused by a large amount of noise leading incorrect initial configurations or random energy level jumps during the annealing process. Both effects occur with growing probability for increasing annealing times.

3.6.2 Scalability. Figure 6 compares scalability of the two reductions by analysing the amount of required physical qubits, and the mean length of chains necessary to connect qubits without direct physical connections (we use the *minorminer* tool provided as part of the D-Wave API to embed QUBOs into the hardware graph).

For the MIS-based method, the amount of physical qubits and mean chain length grow essentially linear with an increasing α, which follows from the pairwise links between conflicting literals.

The backbone method improves upon both aspects because the QUBO is less densely populated, which makes it easier to find embeddings. Especially around the critical value α_c, the amount of required physical qubits is only half of what is required for the MIS method, which in turn implies that substantially larger problem sets can be solved on a hardware of given size.

4 CONCLUSION

Development and evaluation of quantum software components must address well-established engineering concerns of traditional SWE. Based on the scenario of replacing SAT solving, a key element of many applications, with a quantum component, we have shown that careful attention is required in defining and evaluating relevant qualities. We have argued that scalability and accuracy are of particular relevance for early existing quantum annealers. While replacing classical with quantum components is not particularly involved from a programming perspective, our experiments indicate that engineers must be aware of crucial details that might be perceived as irrelevant in traditional SWE to make informed decisions on potentials and pitfalls of quantum computing.

REFERENCES

[1] Alastair A. Abbott, Cristian S. Calude, Michael J. Dinneen, and Richard Hua. 2019. A hybrid quantum-classical paradigm to mitigate embedding costs in quantum annealing. *International Journal of Quantum Information* 17, 05 (2019), 1950042. https://doi.org/10.1142/S0219749919500424 arXiv:https://doi.org/10.1142/S0219749919500424

[2] Saswat Anand, Edmund Burke, Tsong Chen, John Clark, Myra Cohen, Wolfgang Grieskamp, Mark Harman, Mary Harrold, and Phil Mcminn. 2013. An orchestrated survey on automated software test case generation. *Journal of Systems and Software* 86 (08 2013), 1978–2001. https://doi.org/10.1016/j.jss.2013.02.061

[3] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510.

[4] Len Bass, Paul Clements, and Rick Kazman. 2012. *Software Architecture in Practice* (3rd ed.). Addison-Wesley Professional.

[5] Jun Cai, William G Macready, and Aidan Roy. 2014. A practical heuristic for finding graph minors. *arXiv preprint arXiv:1406.2741* (2014).

[6] Peter C Cheeseman, Bob Kanefsky, and William M Taylor. 1991. Where the really hard problems are.. In *IJCAI*, Vol. 91. 331–337.

[7] Vicky Choi. 2008. Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. *Quantum Information Processing* 7, 5 (2008), 193–209. https://doi.org/10.1007/s11128-008-0082-9

[8] Vicky Choi. 2010. Adiabatic Quantum Algorithms for the NP-Complete Maximum-Weight Independent Set, Exact Cover and 3SAT Problems. *ArXiv* abs/1004.2226 (2010).

[9] Thomas Gabor, Sebastian Zielinski, Sebastian Feld, Christoph Roch, Christian Seidel, Florian Neukart, Isabella Galter, Wolfgang Mauerer, and Claudia Linnhoff-Popien. 2019. Assessing Solution Quality of 3SAT on a Quantum Annealing Platform. In *Quantum Technology and Optimization Problems*, Sebastian Feld and Claudia Linnhoff-Popien (Eds.). Springer International Publishing, Cham, 23–35.

[10] Alexander S Green, Peter LeFanu Lumsdaine, Neil J Ross, Peter Selinger, and Benoît Valiron. 2013. Quipper: a scalable quantum programming language. In *Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation.* 333–342.

[11] Thomas Häner, Damian S Steiger, Krysta Svore, and Matthias Troyer. 2018. A software methodology for compiling quantum programs. *Quantum Science and*

Technology 3, 2 (2018), 020501.

[12] T S Humble, A J McCaskey, R S Bennik, J J Billings, E F D'Azevedo, B D Sullivan, C F Klymko, and H Seddiqi. 2014. An integrated programming and development environment for adiabatic quantum optimization. *Computational Science & Discovery* 7, 1 (jul 2014), 015006. https://doi.org/10.1088/1749-4680/7/1/015006

[13] Richard M Karp. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer, 85–103.

[14] Emmanuel Knill. 1996. *Conventions for quantum pseudocode*. Technical Report. Los Alamos National Lab., NM (United States).

[15] Mark W. Lewis and Fred Glover. 2017. Quadratic Unconstrained Binary Optimization Problem Preprocessing: Theory and Empirical Analysis. *ArXiv* abs/1705.09844 (2017).

[16] Andrew Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2 (2014), 5.

[17] Wolfgang Mauerer. 2005. Semantics and simulation of communication in quantum programming. *arXiv preprint quant-ph/0511145* (2005).

[18] Catherine C. McGeoch. 2014. Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice. In *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice*.

[19] Catherine C McGeoch. 2019. Principles and guidelines for quantum performance analysis. In *International Workshop on Quantum Technology and Optimization Problems*. Springer, 36–48.

[20] Andriy V. Miranskyy and Lei Zhang. 2018. On Testing Quantum Programs. *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)* (2018), 57–60.

[21] David Mitchell, Bart Selman, and Hector Levesque. 1992. Hard and easy distributions of SAT problems. In *AAAI*, Vol. 92. 459–465.

[22] Florian Neukart, Gabriele Compostella, Christian Seidel, David Von Dollen, Sheir Yarkoni, and Bob Parney. 2017. Traffic flow optimization using a quantum annealer. *Frontiers in ICT* 4 (2017), 29.

[23] Gerardo Ortiz, James E Gubernatis, Emanuel Knill, and Raymond Laflamme. 2001. Quantum algorithms for fermionic simulations. *Physical Review A* 64, 2 (2001), 022319.

[24] Ferdinand Peper. 2017. The End of Moore's Law: Opportunities for Natural Computing? *New Generation Computing* 35 (2017), 253–269.

[25] Markus Reiher, Nathan Wiebe, Krysta M Svore, Dave Wecker, and Matthias Troyer. 2017. Elucidating reaction mechanisms on quantum computers. *Proceedings of the National Academy of Sciences* 114, 29 (2017), 7555–7560.

[26] Irmi Sax, Sebastian Feld, Sebastian Zielinski, Thomas Gabor, Claudia Linnhoff-Popien, and Wolfgang Mauerer. 2020. Towards Understanding Approximation Complexity on a Quantum Annealer. *Digitale Welt* 4, 1 (01 Jan 2020), 104–104. https://doi.org/10.1007/s42354-019-0244-1

[27] Seung Woo Shin, Graeme Smith, John A Smolin, and Umesh Vazirani. 2014. How" quantum" is the D-Wave machine? *arXiv preprint arXiv:1401.7087* (2014).

[28] Peter W. Shor. 1994. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26 (1994), 1484–1509.

[29] Michael Sipser. 2006. *Introduction to the Theory of Computation* (second ed.). Course Technology.

[30] Damian S Steiger, Thomas Häner, and Matthias Troyer. 2018. ProjectQ: an open source software framework for quantum computing. *Quantum* 2 (2018), 49.

[31] Tobias Stollenwerk, Elisabeth Lobe, and Martin Jung. 2019. Flight gate assignment with a quantum annealer. In *International Workshop on Quantum Technology and Optimization Problems*. Springer, 99–110.

[32] Tobias Stollenwerk, Bryan O'Gorman, Davide Venturelli, Salvatore Mandrà, Olga Rodionova, Hokkwan Ng, Banavar Sridhar, Eleanor Gilbert Rieffel, and Rupak Biswas. 2019. Quantum annealing applied to de-conflicting optimal trajectories for air traffic management. *IEEE transactions on intelligent transportation systems* 21, 1 (2019), 285–297.

[33] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. 2018. Q# Enabling Scalable Quantum Computing and Development with a High-level DSL. In *Proceedings of the Real World Domain Specific Languages Workshop 2018*. 1–10.

[34] Krysta M Svore, Alfred V Aho, Andrew W Cross, Isaac Chuang, and Igor L Markov. 2006. A layered software architecture for quantum computing design tools. *Computer* 39, 1 (2006), 74–83.

[35] Muhammad Tahir, Fazlullah Khan, Muhammad Babar, Fahim Arif, and F Khan. 2016. Framework for better reusability in component based software engineering. *the Journal of Applied Environmental and Biological Sciences (JAEBS)* 6, 4S (2016), 77–81.

[36] Tassio Vale, Ivica Crnkovic, Eduardo Santana De Almeida, Paulo Anselmo Da Mota Silveira Neto, Yguaratã Cerqueira Cavalcanti, and Silvio Romero de Lemos Meira. 2016. Twenty-eight years of component-based software engineering. *Journal of Systems and Software* 111 (2016), 128–148.

[37] Davide Venturelli and Alexei Kondratyev. 2019. Reverse quantum annealing approach to portfolio optimization problems. *Quantum Machine Intelligence* 1, 1-2 (2019), 17–30.

A ALTERNATIVE REDUCTIONS

Loosened Clause Penalties Choi's reduction (definition 3.1), is, in essence, a reduction from k-SAT to the problem of finding the maximal independent set (MIS) of a given graph. Assume a k-SAT instance is reduced to QUBO as described in definition 3.1, and let G_f be the graph representation. Consider a MIS of G_f, which is given by the largest set of vertices such that there are no connected vertices. In definition 3.1 this property is enforced by the constraint $\delta > \omega$. Solving a QUBO defined by definition 3.1 also solves the MIS problem for G_f. The problem of finding the MIS G_f corresponds to the problem of finding the maximal number of satisfiable clauses (MAX-k-SAT) in f. The relation between MAX-k-SAT and k-SAT is trivial.

THEOREM A.1. *Setting $\delta = \omega$ in definition 3.1 does not change the correctness of the assignment derived from the QUBO solution.*

PROOF. Let $q(x)$ be a sub-QUBO representing one clause like described in definition 3.1. Under $\delta = \omega$ the following holds: $\min(E(n)) = -\omega$ if $E(n)$ is the energy of a clause sub-QUBO with n satisfied literals. It is straight forward to see that $E(n) = -n\omega + \binom{n}{2}\omega$. Therefor, $E(0) = 0$ and $E(1) = E(2) = -\omega$. The inequality $E(n) < E(n+1)$ evaluates to $-n < -1$ which is true for all $n > 1$. This leads to the conclusion that $\min(E(n)) = -\omega$.

Consider a clause C_i and its corresponding sub-QUBO $q_i(x)$. Now, $\min(q_i(x)) = -\omega$ for one or two satisfied literals in $q_i(x)$. Therefore, the minimization of $q_i(x)$ leads to a satisfied clause C_i. For two conflict-free clauses C_i and C_j the combined minimum energy is given by $\min(q_i(x) + q_j(x)) = -2\omega$. Now we introduce a conflict between C_i and C_j. That activates an additional penalty term $p_{ij} = \omega$ which leads to $\min(q_i(x) + q_j(x) + p_{ij}) = -\omega > -2\omega$. This shows that conflicts between clauses always lead to a higher energy level and thus should be avoided when minimizing the complete QUBO $q(x)$. For all satisfiable k-SAT instances $f(x)$ with n clauses the minimal energy of their corresponding QUBOs $q(x)$ will be $min(q(x)) = -n\omega$. Every function $f(x)$ with minimal QUBO value $\min(q(x)) > -n\omega$ cannot be satisfied. □

Backbone Choi's reduction represents variables solely by their literals. To avoid conflicts, we need to ensure that $l_{ij} \neq l_{i'j}$ for all pairs $(l_{ij}, l_{i'j}) \Leftrightarrow (x_j, \bar{x}_j)$. An edge in the QUBO connects the literals as penalty term, which leads to highly connected graphs for instances with large values of α. The degree of connectivity can be improved by introducing a *backbone* for variables, which allows us to transitively express equivalence between literals by linking them to their corresponding variable. The reduction is given in Definition 3.2 on page 4.

CORRECTNESS OF DEFINITION 3.2. Let $E(n)$ be the energy of a clause sub-QUBO with n satisfied literals. The difference between sub-QUBOs in definition 3.1 and definition 3.2 is that in the latter node weights of literals $l_{ij} \neq x_j$ are moved to the edges (l_{ij}, x_j). For every literal, there exists exactly one edge to its corresponding variable. Therefore, edge weights can be viewed as node weights, and it follows that $E(n) = -n\omega + \binom{n}{2}\omega$. Consequently, $\min(E(n)) = -\omega$ also holds for definition 3.2. If two literals l_{ij} and $l_{i'j}$ conflict, one of $\omega l_{ij} x_j$ or $\omega l_{i'j} x_j$ evaluates to ω, while the other evaluates to 0. The rest of the argument follows Theorem A.1. □