A comprehensive benchmark of an Ising machine on the Max-Cut problem

To cite this article before publication: Salwa Shaglel *et al* 2026 *New J. Phys.* in press https://doi.org/10.1088/1367-2630/ae7823

A comprehensive benchmark of an Ising machine on the Max-Cut problem

Salwa Shaglel*†[1], Markus Kirsch‡[2], Marten Winkler[1], Christian Münch[2], Stefan Walter[2], Fritz Schinkel[2], Martin Kliesch[1]

[1]Institute for Quantum Inspired and Quantum Optimization, Hamburg University of Technology, Blohmstraße 15, 21079 Hamburg, Germany
[2]Fujitsu Germany GmbH, Mies-van-der-Rohe-Straße 8, 80807 Munich, Germany

QUBO formulations of combinatorial optimization problems allow for solving them using various quantum heuristics. While large-scale quantum computations are currently still out of reach, we can already numerically test such QUBO formulations on a perhaps surprisingly large scale.

In this work, we benchmark Fujitsu's Digital Annealer (DA) on the Max-Cut problem, which captures the main complexity of the QUBO problem. We make a comprehensive benchmark against leading other heuristic algorithms on graphs with up to 53,000 variables by focusing on the wall-clock time. Moreover, we compare the DA performance against published performance results of the D-Wave hybrid quantum-classical annealer and the recently proposed QIS3 heuristic. Based on performance statistics for over 2,000 graphs from the MQLib, we find that the DA yields competitive results. We hope that this benchmark demonstrates the extent to which large QUBO instances can be heuristically solved today, yielding consistent results across different solvers.

Contents

*corresponding author
†salwa.shaglel@tuhh.de
‡markus.kirsch@fujitsu.com

1. Introduction

Combinatorial optimization problems are ubiquitous in many applications, including vehicle routing problems, product assembly optimization, finance portfolio optimization, RNA folding problems, machine learning, and numerous other domains [1]. Often, they can be formulated as quadratic unconstrained binary optimization (QUBO) problems [1–4], which naturally fit certain quantum computing approaches. Since quantum hardware is not sufficiently mature yet to solve these problems at practically relevant scales, often classical QUBO solvers are used to solve large QUBO problems. Therefore, the QUBO solvers are often called *quantum-inspired* even though the underlying basic methods, such as simulated annealing and parallel tempering, predate the quantum computing efforts.

Indeed, in the current absence of sufficiently powerful quantum hardware, it is difficult to assess the performance of a quantum heuristic. At the same time, it is important to single out problems where the QUBO approach is particularly promising. This information can be essential for improving the quantum-readiness of those problems. It is highly desirable for QUBO formulations of practical use-cases to be tested today at as large a scale as possible to gain insight into how potential quantum computing approaches might solve relevant QUBO problems. Current quantum hardware can operate with an

order of 1000 noisy qubits. In contrast, dedicated QUBO solvers can deal with tens of thousands of bits without suffering from noise and severe connectivity constraints typically encountered in quantum hardware.

The QUBO problem is NP-hard [5], and as such, it is not expected to be solvable in polynomial time, even with quantum computers [6]. This fundamental limitation has driven recent research toward the development of heuristic algorithms and the design of specialized classical and quantum hardware. These approaches intentionally trade off optimality guarantees in favor of scalability and speed, aiming to tackle intractable problems in practice. Notable examples include the simulated bifurcation machine by Toshiba [7, 8], the quantum annealer by D-Wave [9], Hitachi's Momentum Machines [10], NTT's simulated coherent Ising machines [11, 12], and the digital annealer by Fujitsu [13]. The goal of this work is to shed light on the current capabilities of such a so-called *Ising machine* [14]–the Digital Annealer (DA). The DA is a CMOS-based ASIC designed to efficiently execute an enhanced simulated annealing algorithm. By some it is referred to as 'quantum-inspired' because it solves a class of problems that is native to certain quantum computing approaches.

Among the many QUBO-encoded problems, the maximum cut (Max-Cut) problem stands out as a natural NP-hard problem that admits a straightforward QUBO formulation. This ease of formulation, in addition to its relevance across graph theory and combinatorial optimization, has made Max-Cut a popular benchmark for testing and comparing optimization solvers. Goemans and Williamson [15] introduced a polynomial-time randomized approximation algorithm based on semidefinite programming, achieving a worst-case performance guarantee (approximation ratio) of at least 0.878. Assuming the unique games conjecture and $\mathsf{P} \neq \mathsf{NP}$, that is the best value that can be achieved in polynomial time [16]. In contrast to approximation algorithms, as of 2018 [17], over 95 heuristics have been proposed for Max-Cut. These heuristics prioritize computational efficiency and practical performance over guarantees and are widely employed in large-scale applications where exact or approximate algorithms become infeasible.

In this work, we benchmark and compare different algorithms and hardware. In order to have a useful and meaningful benchmark, it must be set up, executed, and reported on as transparently as possible. Without such transparency, a benchmark can lack rigor, fairness, or even become misleading [17, 18]. In particular, besides the general benchmark setup and hardware specifications, the following information should be provided:

1. **Benchmark set**: (1a) Convincing reasons for the selection of the chosen benchmark set. (1b) The set should reflect the diversity and complexity of real-world problem distributions. In particular, (1c) it should not be chosen due to solver-dependent considerations.

2. **Runtimes**: (2a) Convincing arguments for the chosen runtime limits, and (2b) deviations from those limits; (2c) tuning times and to what extent they have been included in the runtime. Not fully including tuning times in the runtime should be convincingly justified.

Spec. \ Solvers		MQLib heuristics [17]		D-Wave HS [19]	QIS3 [20]	
		DAv2	DAv3	DAv3	DAv2	DAv3
Instances	amount	2,044	819	45	14	16
	size	[200 – 8,176]	[2,048 – 53,130]	[2,000 – 10,000]	[800 – 8,000]	[800 – 10,000]
	density	[0.00032 – 1]	[0.0001 – 0.995]	[0.0004 – 0.97]	[0.0005 – 0.06]	[0.0004 – 0.06]
Library		MQLib [21]		MQLib	G-set [22]	
Time limit		instance-specific		20 minutes	10 seconds	

Table 1: Summary of benchmarks considered in this work. For the benchmark on the MQLib heuristics, we restrict our benchmark to those instances that can be reliably solved within the assigned time and that fit the chip (For transparency, the raw data of the unanalyzed instances are also available on GitHub [23]). For the benchmarks against D-Wave's HS and QIS3, we follow their original selections and use their provided data.

3. **Performance details**: (3a) Convergence speeds, (3b) number of trials, (3c) parallelization of the algorithm, and (3d) the obtained solutions (not just the objective values) so that independent verification is possible.

Our study follows all of these desiderata by conducting a systematic and fair enough comparison of the second- and third-generation DA, referred to as DAv2 and DAv3, respectively, on a broad and diverse set of Max-Cut instances. Throughout the methodology and results sections, we refer back to these points to indicate where and how each criterion has been addressed.

First, we compare the DA's performance against the known best classical heuristics from MQLib. To this end, we build on a benchmark of 37 heuristics [17] by selecting the most powerful ones for our comparison. More specifically, we choose those heuristics that demonstrated the best performance across different combinations of instance size and density. Our benchmark covers a total of 2,125 Max-Cut instances. To promote fairness and comparability, we adopt the methodology of Ref. [17], which assigns a tailored time limit to each instance based on its computational difficulty and accounts for expected hardware improvements over time.

To complement this benchmark, we compare the performance of DAv3 with D-Wave's hybrid quantum-classical solver on their selection of 45 Max-Cut instances [19], as well as with the recent quantum-inspired metaheuristic QIS3 solver [20] on their selection of 16 instances. In both comparisons, we shed light on DA's rapid convergence properties. A summary of the benchmarks performed in this work is presented in Table 1.

1.1. Previous work

Quantum computing has introduced several paradigms aimed at solving combinatorial optimization problems such as Max-Cut. One of the most studied gate-based quantum approaches is the quantum approximate optimization algorithm (QAOA), introduced by Farhi *et al.* [24]. QAOA provides approximate solutions for combinatorial optimization problems using a variational

circuit. It has been specifically evaluated on 2-regular and 3-regular Max-Cut instances with single-layer depth, achieving a theoretical approximation ratio of at least 0.6924. This ratio is expected to improve with increasing circuit depth, as proven in [25]. For a detailed overview of QAOA and its extensions, see [26]. Despite its potential, QAOA remains in its early stages of development. Demonstrating consistent quantum advantage with QAOA is currently out of reach and remains an active area of research due to several ongoing challenges [26]. These include: finding tractable problems for quantum computers [27, 28], and the complexity of parameter optimization as the circuit depth increases due to barren plateaus [29], local minima [30] and hardness of finding hyperparameters [31]. Moreover, the impact of hardware noise [32, 33] and overheads due to noise reduction [34, 35] pose additional challenges.

In parallel to gate-based quantum computation, a distinct approach to solving the Max-Cut problem has emerged through quantum annealing and related Ising machines. Quantum annealers, such as those built by D-Wave Systems, are a prominent hardware realization of this approach [36]. The state-of-the-art quantum annealer is the D-Wave Advantage2 quantum processing unit, which contains over 4,400 superconducting qubits and utilizes the Zephyr topology with degree-20 connectivity [9]. This marks a substantial improvement over the earlier Pegasus-based Advantage system [37]. In a recent benchmark reported in D-Wave's white paper [9], both Advantage and Advantage2 were used to solve the largest 3D-lattice spin glasses, which can be embeddable in their respective topologies—a $12 \times 12 \times 12$ cube with 1,650 variables and 4,461 edges. The Advantage2 system consistently returned better-quality solutions under identical annealing times and, in many cases, outperformed the Advantage system by several orders of magnitude in speed. Nonetheless, despite these technological advances, current quantum annealers face several critical limitations. These include restricted qubit counts, limited connectivity, and short coherence times [38]. As the number of qubits increases, so does the complexity of coupler layout, introducing more noise—especially when many couplers are redundant for a given problem instance [39, 40]. These constraints pose serious challenges for the scalability and universality of quantum annealers.

Given the current limitations of quantum hardware, quantum-inspired approaches offer a practical alternative for tackling combinatorial optimization problems. Building upon the previously reviewed hardware considerations, recent studies have sought to empirically benchmark the performance of emerging quantum and quantum-inspired annealing platforms. Notably, Fujitsu's DA has been evaluated against both classical heuristics and quantum annealing systems.

Matsubara *et al.* [41] provide an evaluation of DAv2, using 65 instances from the G-set benchmark suite [22]. These instances, which span sparse graphs with edge densities between 0.0002 and 0.06 and variable counts ranging from 800 to 8000, were compared against results obtained using IBM's CPLEX solver by Ref. [42] and a classical Multiple Operators heuristic [43]. The study finds that DAv2 achieves the best cut values for 62 out of the 65 instances and delivers the shortest runtime for 52 of them, underscoring its efficiency

on sparse yet moderately sized Max-Cut problems. Our work improves upon this study by addressing several of its limitations, specifically those related to (1b), (2a), (3a), and (3d) from the benchmarking criteria list.

A more nuanced comparison is provided by Huang *et al.* [44], who evaluate both the D-Wave quantum annealer and Fujitsu's DAv2 across 10 Max-Cut instances ranging from 543 to 5430 variables and constrained to Pegasus- or Chimera-like graphs to conform to the architecture of D-Wave's quantum processing unit. Their results show that Pegasus-based and Chimera-based quantum annealers (used in earlier D-Wave systems) perform well on smaller or sparsely connected graphs but show diminishing results on larger or denser instances in comparison to DA. To further examine the dependence of solver's performance on graph structure, Huang *et al.* [44] generated 32 Max-Cut instances, each with 145 variables and average degrees ranging from 1 to 140. While these instances were still more or less tailored to D-Wave's quantum processing unit connectivity, the DA demonstrated improved performance compared to the quantum annealer. This study does not meet several important criteria from the list, particularly (1b), (1c), and (3d). Although the study explicitly discusses hyperparameter tuning, it excludes the tuning process from the reported runtime, thereby not fully including (2c).

Complementing the comparative studies discussed earlier, Oshiyama and Ohzeki [45] conducted an extensive benchmark involving multiple advanced solvers: D-Wave's hybrid solver (HS), Toshiba's Simulated Bifurcation Machine (SBM), Fujitsu's DAv2, and simulated annealing. Their evaluation focused on the Max-Cut problem using 45 instances (chosen by D-Wave [19]) from the Max-Cut and QUBO instances library (MQLib), restricted to problem sizes up to 8000 variables. Their findings indicate that DAv2 outperformed the other solvers on large instances, although it did yield slightly inferior results on a few specific cases. Aggregated, D-Wave's HS achieved the best results on 22 out of the 45 instances, followed by DAv2 (20), SBM (16), and simulated annealing (7). When broken down by problem size, D-Wave's HS led on small instances, while DAv2 dominated on medium and large ones. In terms of edge density, D-Wave's HS was most effective on sparse graphs, whereas DAv2 performed best on graphs with medium to high connectivity. Our study extends this comparison by benchmarking DAv3 against D-Wave's HS, while addressing important gaps in notably (1b), (2a), (3a), and (3d).

Further evidence of DA's competitive performance is presented in a separate benchmarking study by Şeker *et al.* [46]. This study compares DAv2 to three exact solvers—GUROBI [47], CPLEX [42], and SCIP [48]—as well as two classical heuristics: BURER2002 [49] and PAL2004bMTS2 [50] as implemented by Dunning *et al.* [17]. Using a sample of 260 Max-Cut instances with up to 8000 vertices, and runtime limits of 60 and 120 seconds, the results show that DAv2 consistently yields solution quality and runtimes that are better or competitive, further reinforcing its promise for practical, time-constrained optimization. However, there is still potential to include, in particular, the criteria (2a), (2b), and (3d). While some aspects of (3a) are briefly mentioned, we provide concrete evidence to support it.

Jiang *et al.* [51] benchmark three hardware-based annealers—D-Wave Advantage, Fujitsu's DAv3, and Quantix GPUA—alongside a classical solver from Meta-Analytics, across several NP-hard problems including Max-Cut. For Max-Cut, they evaluate 10 instances with 2000 vertices at varying densities. Despite the limited sample, DAv3 consistently finds the Max-Cut across all instances and yields better results than other hardware solvers in both solution quality and runtime across all problems considered. Several important aspects are not covered by this study, including (1a), (1b), (2a), (3a), and (3d).

Beyond Max-Cut, Fujitsu's DA has been applied successfully to various NP-hard combinatorial optimization problems reformulated as QUBO models. Studies have explored the DA's performance on graph problems such as 3-Regular 3-XORSAT [52], quadratic assignment [41, 44, 46], minimum-cut [41], 3-SAT [53], NAE 3-SAT [45], number and graph partitioning [54], and minimum vertex cover [44]. Additionally, the DA has been used effectively in scientific domains including transport robot scheduling [55], molecular structure optimization [56], magnetic phase discovery [57], and 2-D magnetic array design for energy harvesting and planar motors [58]. These studies generally report favorable solution quality and time-to-solution for the DA. Moreover, QUBO modeling facilitates integration into multi-phase solution strategies. For instance, Dornemann *et al.* [59] employs the DA in the final phase of a three-step approach to the capacitated vehicle routing problem with time windows, using it to solve a set partition problem and obtain a feasible global solution.

1.2. Our contribution

We aim to answer the question of how QUBO solver performances compare from a user perspective. Therefore, we focus on the wall-clock time and compare the obtained objective values.

While several previous studies already provide some benchmarks, we put an emphasis on following all the desired benchmarking standards on 1. the benchmark set selection, 2. the runtime choices, and 3. reporting performance details (see Section 1).

Our main study includes a diverse subset of heterogeneous instances from MQLib, representing the largest test suite among all works reviewed in Section 1.1 (1b). Following Dunning *et al.* [17], we obtain instance-specific baseline time limits determined as the convergence time of a simple greedy local search heuristic (2a). Each heuristic (DAs and MQLib) is then run individually on the same server to avoid interference from background load or parallel processes. By establishing these updated time limits, rather than reusing those from Ref. [17], we take the hardware specifications into account and assign proper time limits that capture the computational complexity of instances. This avoids both overestimating and underestimating solver performance.

This sets the stage for our first selection of instances from MQLib. We first

exclude all instances with time limits below 0.25 seconds, as these are considered too easy and prone to the solver's runtime overshooting. The second filtering criterion is based on size (number of variables). A single Digital Annealer Unit (DAU) supports instances up to 8,192 variables, so for DAv2 we consider instances up to this size. In contrast, DAv3 has an additional software layer capable of handling instances of up to 100,000 variables. However, we restrict our analysis to instances with at least 2,048 variables, as we observed significant runtime overshooting on smaller instances, making the comparison less meaningful (1a). Importantly, the remaining instances are not selected based on the DA's considerations (e.g., connectivity, edge-weight distribution or type, structural features) to avoid bias toward instances known to favor DA performance (1c). The details of this benchmark set is presented in Table 1.

In addition, we report actual runtimes rather than just the time limits and explicitly analyze deviations from these limits (2b). We also include in the total runtime the overhead from the DA's automatic internal hyperparameter tuning, and avoid any further manual parameter tuning (2c).

We emphasize that the computational architectures we consider differ significantly in nature. The DA is a highly parallelized heuristic (3c), whereas the classical baseline heuristics from MQLib are single-threaded solvers. The aim of this work is not to modify the solvers but to evaluate them as provided, enabling us to observe performance differences across computational architectures. Indeed, our results indicate that for certain instances, a single-threaded classical heuristic solver can outperform a parallelized dedicated one. Nevertheless, our direct comparison in terms of wall-clock time naturally favors a parallel architecture. Parallelized or GPU-based classical solvers (e.g., multi-threaded simulated annealing, parallel tempering, or modern GPU heuristics) are not included in the present study. A review of available implementations did not yield candidates that are publicly available, well-documented, and reproducible solvers to be integrated into a large-scale and reproducible benchmark as we aim in this work. This consideration was the main motivation for including additional complementary benchmarks with D-Wave's HS and QIS3.

For the benchmarks involving D-Wave's HS and QIS3, we use the benchmark sets and runtime limits provided in the papers [19, 20] Therefore, we deviate from some of our benchmarking criteria, particularly (1a), (1b), potentially (1c), and (2a), as these aspects are shaped by how the original solver studies were conducted. However, we still ensure the criteria (2b), (2c), (3a), (3b), and (3d), thereby providing as rigorous and transparent a comparison as currently feasible. We particularly analyze the convergence behavior of DAv3 (3a).

For all solvers executed in this work, we performed 5 runs with different seeds and perform the benchmark on the best objective value among them (3b). The best-of-5 summary reflects an application-driven setting in which users can afford only a small number of restarts and select the best outcome. In addition, whenever applicable, we provide statistical inference using majority-based paired run-level significance tests to rigorously evaluate whether observed differences between solvers are statistically meaningful rather than due

to random variation. Additional report of robustness statistics across seeds (e.g., median and standard deviation) and the solution configurations (binary variable vectors) for all instances involved in this study are made available in our Git repository [23] (3d).

The rest of this paper is organized as follows. Section 2 provides the necessary background, including an overview of QUBO, the Max-Cut formulation, and a fundamental introduction to the DA. The detailed methodology of our benchmarks against classical heuristics, D-Wave's HS, and QIS3 is described in Section 3. Finally, we present and discuss the performance results in Section 4.

2. Background

This section provides a general introduction to QUBO problems, the Max-Cut problem, and the DA, along with the basic definitions and notations used throughout this work.

2.1. QUBO Problems

Many combinatorial optimization problems can be formulated as a QUBO problem given as

$$\begin{aligned} &\text{minimize } \mathbf{x}^\top Q\mathbf{x} = \sum_{i\geq j}^{n} q_{ij}x_i x_j, \\ &\text{subject to } \mathbf{x} \in \{0,1\}^n, \end{aligned} \tag{1}$$

where $Q \in \mathbb{R}^{n\times n}$ is a symmetric matrix with diagonal elements q_{ii} corresponding to linear terms (since $x_i x_i = x_i$ for $x_i \in \{0,1\}$), while the off-diagonal elements q_{ij} for $i \neq j$ correspond to quadratic terms. The goal is to find an optimal solution configuration $\mathbf{x}^* \in \{0,1\}^n$ that minimizes the quadratic objective function (1). Constrained models can also be reformulated as QUBO problems by introducing penalty terms into the objective function, instead of explicitly imposing constraints. These penalty terms are typically constructed to be zero for feasible configurations and positive for infeasible ones. A penalty prefactor is typically included to control the cost of constraint violation, and its value must be appropriately chosen to ensure the resulting solution is both optimal and feasible [3].

In fact, Eq. (1) is directly related to finding the ground state of the Ising Hamiltonian $-\sum_{i=1}^{n} h_i s_i - \sum_{i>j}^{n} J_{ij} s_i s_j$, where spin variables $\mathbf{s} \in \{-1,1\}^n$ relates to $\mathbf{x} \in \{0,1\}^n$ by $s_i = 2x_i - 1$. In this formulation, the coefficients J_{ij} represent the interaction strength between spins s_i and s_j, and h_i corresponds to the strength of the external magnetic field acting on spin s_i. This connection underlines the naming of many hardware solvers designed for such problems as Ising machines [14].

The QUBO problem is NP-hard. Therefore, all QUBO solvers are expected to exhibit a worst-case runtime that scales exponentially with the size of the

problem n, also for quantum solvers. That is, general large instances are expected to be computationally intractable. However, for commonly chosen instances and tolerable approximation errors, the scaling might be more favorable. Many NP-hard problems have already been formulated in the QUBO form, with 112 such formulations listed in Ratke's blog [4].

The Max-Cut problem

Given a weighted undirected graph with a vertex set V, edge set E, and symmetric weight matrix W, a cut in this graph is a set of edges with exactly one end in a related $S \subset V$:

$$\delta(S) = \{\{v_i, v_j\} \in E |\ v_i \in S \text{ and } v_j \in \bar{S}\}, \tag{2}$$

where $\bar{S} = V \setminus S$ is the complement set of S. The Max-Cut problem is to find a cut of V into two disjoint subsets $S \subset V$ and $\bar{S}$ with maximum sum of edge weights w_{ij} between them, i.e., $f(S) = \sum_{\{v_i,v_j\}\in\delta(S)} w_{ij}$ is maximized. We will refer to $f(S)$ as the *objective value* or the *cut value*, which will be used interchangeably throughout this paper. We call an instance *unweighted* if all edges have weight one, i.e., $w_{ij} = 1$ if $\{v_i, v_j\} \in E$ and $w_{ij} = 0$ otherwise. In this case, the objective is to maximize the number of edges in the cut, and thus the cut value is then given by $f(S) = |\delta(S)|$.

The Max-Cut problem is directly reducible to a QUBO problem with little overhead, making it a representative problem that captures the main computational challenges of QUBO problems. As such, any Max-Cut problem on a graph can be expressed as in Eq. (1) with $|V|$ binary variables. Every vertex $v_i \in V$ in a graph corresponds to a binary variable $x_i \in \{0, 1\}$ in the QUBO problem. We assign the variable x_i to 1 if $v_i \in S$ and 0 otherwise. For an edge between v_i and v_j to be in the cut $\delta(S)$, it must thus hold $x_i \neq x_j$. Substituting the coefficients of the QUBO formula with $q_{ij} = -2w_{ij}$, and $q_{ii} = 2\sum_{j\in N(i)} w_{ij}$, where $N(i)$ denotes the set of neighbors of vertex x_i, we can write the Max-Cut problem in its QUBO formulation

$$f(\mathbf{x}) = \sum_{\{i,j\}\in E} w_{ij}(x_i + x_j - 2x_i x_j), \tag{3}$$

where $\{i, j\}$-summand evaluates to 1 if x_i and x_j are different and 0 otherwise. In particular, maximizing Eq. (3) is equivalent to solving the Max-Cut problem. Since, by convention, Ising machines minimize rather than maximize, the equivalent problem is to minimize $-f(\mathbf{x})$.

The equivalence also implies that any QUBO instance can be formulated as a Max-Cut instance in a graph with $|V| + 1$ vertices. While coefficients of quadratic terms in the QUBO problem correspond directly to edge weights between vertices in a graph, linear terms do not have a natural graph representation. However, they can be embedded into the graph structure by introducing an auxiliary vertex x_0, connecting it to all other vertices in the graph, and fixing its value to 1. This allows linear terms to be interpreted as additional edge weights incident on x_0. The mapping of QUBO formulation

(1) to the Max-Cut problem (3) can be shown mathematically by assigning the coefficients as $w_{ij} = -\frac{1}{2}q_{ij}$ and $w_{0j} = 2\sum_{\substack{i>0 \\ i\in N(j)}} w_{ij} - q_{jj}$.

2.2. The Digital Annealer

The DA is a technology specifically designed to heuristically address combinatorial optimization problems formulated in QUBO form. Its core component, the DAU, is an application-specific integrated circuit (ASIC) constructed using CMOS technology, capable of directly realizing QUBO problems with up to 8,192 binary variables. The DAU employs a simulated annealing variant as a basic search algorithm, and is directly compatible with any connectivity. This flexibility allows running the DA natively on arbitrary graphs, in contrast to hardware architectures that require native graph topologies or special graph embeddings.

The simulated annealing algorithm is based on a Markov chain Monte Carlo (MCMC) method, namely the Metropolis-Hastings algorithm. It begins with an initial bit-string configuration $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ at a high temperature T, which is a hyperparameter of the algorithm. At each update step, a candidate configuration $\mathbf{x}' \in \{0,1\}^n$ is generated by flipping a randomly chosen variable $x_i \in \mathbf{x}$ for some $i \in \{1, 2, 3, \ldots, n\}$ and the energy difference $\Delta E = E(\mathbf{x}') - E(\mathbf{x})$ is computed. The new configuration is accepted with probability

$$p = \min\{\exp(-\Delta E/T), 1\}. \tag{4}$$

This choice implies that if $\Delta E < 0$, the new configuration is always accepted, while if $\Delta E > 0$, it is accepted with probability $\exp(-\Delta E/T)$. This probabilistic acceptance allows the algorithm to escape local minima by occasionally accepting worse configurations. As the temperature gradually decreases according to a suitable cooling schedule [60], this probability diminishes, reducing exploration and encouraging convergence towards a local minimum.

In the DA, candidate configurations and their corresponding energy differences are computed in parallel for all variables $x_i \in \mathbf{x}$ where $i \in \{1, 2, 3, \ldots, n\}$. From the set of accepted candidates, one configuration is randomly selected to proceed to the next iteration. This mechanism approximates a rejection-free [52, 61, 62], roulette-wheel-style selection process [63]. In a rejection-free MCMC algorithm, every step leads to a state transition. The DA further enhances the approximate rejection-free simulated annealing algorithm through three modifications:

- *Parallel search*: DAv2 performs up to 128 independent simulated annealing runs that are time shared on 16 parallel threads of a single DAU (see Appendix D). DAv3 uses a single DAU together with a global search on multicore CPUs in parallel.

- *Dynamic energy offsetting* applies an offset to the energy difference, i.e., $\Delta E - \delta E_{\text{off}}$, to increase the acceptance probability in cases where no candidate configurations are accepted, particularly at low temperatures.

This helps escape local minima, provided the offset is not excessively large, to avoid losing the progress of the annealing.

- *Parallel tempering*, also known as replica exchange: This technique runs multiple independent searches at different temperatures in parallel and exchanges configurations among them to improve energy landscape exploration.

In any case, among all assignments evaluated during the algorithm, one with the best objective value is reported as a solution.

The DAU is an integer machine, meaning it does not operate natively on floating-point values. When a QUBO matrix contains floating-point coefficients, an automatic or manual conversion process must be applied. This involves systematically scaling and rounding the matrix elements to produce integer coefficients that fall within the DAU's supported precision range. Specifically, the DAU supports 64-bit signed integers for quadratic terms[1] and up to 76-bit signed integers for linear terms. For DAv2, the precision is reduced to a 16-bit integer when the problem size exceeds 4,096 variables. It offers both simulated annealing (default) and parallel tempering modes.

The ideas of DAv2 are scaled up in DAv3, based on a hybrid approach that integrates a global tabu search in a software layer with the DAU. This combination enables the handling of fully connected QUBO instances with up to 100,000 variables. This version offers parallel tempering only.

DA usage is straightforward; The user needs first to convert their problem into a QUBO formulation (see the tutorial [3]) and then a few lines of code are required to solve it and report the solutions, along with additional details (i.e., energy, time, etc.), see the documentation [23, 64]. Fujitsu's DA can be easily accessed through a cloud platform or on-premises as a service.

3. Methodology

We compare the performance of DAv2 and DAv3 against the best-performing classical heuristics selected from a pool of 37 algorithms implemented by Dunning *et al.* [17]. The heuristics are accessible from the corresponding GitHub repository [21] and used in their defaults. We will refer to these classical heuristics as MQLib heuristics. We also use their collection of heterogeneous problem instances, which includes both real-world and random problem instances provided in [21].

To better understand the strengths of each solver and ensure a reasonable comparison, we split the instance set into 15 groups based on size and density. This grouping allows us to evaluate performance across different problem characteristics and select the top MQLib heuristic performers for each category to compare against. The criteria for splitting and selecting the instance set and

[1] The valid range is $[-2^{63}+1, 2^{63}-1]$.

the selection of the best-of-37 heuristics are described in Sections 3.2 and 3.3, respectively. To ensure a representative comparison, we adopt the time-limit assignment methodology from Ref. [17], which assigns instance-specific time limits reflecting both problem difficulty and hardware improvements over time (Section 3.1).

We include complementary comparisons of DAv3 with D-Wave's hybrid quantum-classical solver [19], as well as comparisons of both DAv2 and DAv3 with the recent quantum-inspired metaheuristic QIS3 [20]. We include the details of their methodology within the relevant subsections. While these additional benchmarks are equally important, their setups follow that of the original studies and thus require less detailed methodology discussion.

3.1. Baseline time limit

Dunning *et al.* [17] suggested assigning a specific time limit to each instance, which reflects its computational difficulty. Harder instances are assigned longer time limits based on characteristics such as size and structure, thereby accounting for the varying effort required to reach high-quality solutions. By doing so, the benchmark avoids both underestimating the performance of solvers on difficult problems (by giving them too little time) and overestimating performance on easier problems (by giving unnecessarily long runtimes). Furthermore, more recent hardware typically requires shorter runtimes on average due to improved processing power compared to older systems. As such, baseline time limits should be periodically updated to reflect current hardware performance. To make the comparison between the DA and the MQLib heuristics on the Max-Cut instances as fair as possible, we follow Ref. [17] by determining new instance-specific baseline time limits (2a).

The instance-specific time limit is determined for each instance by the amount of time required to complete the following local search procedure:

1. Generate a random configuration for the input instance.
2. In each updating step, flip the first occurring variable that improves the configuration (i.e., yields a better objective value).
3. Repeat step 2 until no improving flips can be made.
4. Repeat steps 1, 2, and 3 for 1500 times.

We executed this algorithm on a Fujitsu PRIMERGY RX2540 M5 server with 2x Intel Xeon Gold 6230 CPU @ 2.10GHz (40 cores total), 12 x 32 GB = 384 GB RAM, running RedHat 7.9. Due to the algorithm's inherent randomness, the required runtime may vary across different seeds. Therefore, for each instance, we run this algorithm using 5 different seeds and define the time limit as the mean, or 0.001 seconds if the mean falls below this threshold.

DAv2 has no time limit stopping criterion and instead terminates when the specified number of runs and iterations are completed. Therefore, to set a desired runtime for it, we use a runtime estimate based on the number of variables, runs, and iterations. The number of runs is the number of stochas-

tically independent parallel runs on the chip. The minimum is 16 runs and the maximum is 128 runs for DAv2. The number of iterations is the length of the annealing random walk per run. While the total runtime of DAv2 includes several components, it is predominantly determined by the annealing time and the CPU time, which together account for the majority of the runtime. The CPU time refers to the total time spent by the host CPU on tasks such as communicating with the DAU and preparing its setup. The annealing time is the actual time spent by the DA searching the solution space and finding the optimal or near-optimal solution. Appendix D provides the detailed procedure and fitted runtime models used to estimate the target runtime of DAv2.

In contrast, DAv3 includes a stopping criterion based on a user-specified time limit. However, as is the case with most solvers, DAv3 does not terminate precisely when the specified time limit is reached, as it takes additional time to complete the current search procedure before stopping. Hence, the runtime tends to overshoot. This issue particularly happens for smaller instances, which are typically assigned very short time limits. Notably, DAv3 has a minimum runtime of 1 second; if the specified limit falls below this value, the annealing process may not initiate at all. To address this issue, we incorporate a buffer time for DAv3, aiming to keep the runtime for as many instances to deviate by no more than 10% from the baseline time limit. For instances assigned a time limit below 1 second, we set the minimum to 1 second. We modify the instance-specific time limit T_i^{limit} for DAv3 such that the runtime stays within the margin of 10%:

$$T_i^{\text{limit}} := \max\{1, \min\{\lfloor T_i^{\text{baseline}} \rfloor, \lfloor T_i^{\text{baseline}} * 1.1 - T_i^{\text{offset}} \rfloor\}\}, \tag{5}$$

where T_i^{baseline} is the baseline time limit for instance i, and T_i^{offset} is the buffer time, and it is empirically chosen to be 3 seconds. Our choice of the offset buffer time is described in Appendix E. In our analysis, we favor staying below the baseline time limit rather than exceeding it to keep the comparison to other heuristics as fair as possible. Appendix F presents an analysis of each solver's deviation from the baseline time limit (2b). Notably, we observed that the MQLib heuristics often exceed their specified time limits–an issue not reported in Ref. [17]. Since neither the DA nor the heuristics terminate exactly at the assigned time limit, we allow for only very few instances to slightly exceed the predefined safety margin.

As part of the complementary comparisons in our study, we ensure a consistent basis for comparison with D-Wave's HS and QIS3, and we adhere to their fixed time budgets of 20 minutes and 10 seconds per instance, respectively. We could not find the reasons why these specific values have been chosen in the respective papers.

3.2. Selection of graph instances

Following the benchmark criteria 1 for the benchmark set, we explain and justify below our choice of benchmark set for our main benchmark with MQLib heuristics and provide details of the benchmark sets of the additional benchmarks with D-Wave's HS and QIS3, as selected by the original works.

Size / Density	**x-Small** $20 \leq n < 1,024$	**Small** $1,024 \leq n < 2,048$	**Medium** $2,048 \leq n < 4,096$	**Large** $4,096 \leq n < 8,192$	**x-Large** $8,192 \leq n$
Sparse $d < 0.1$	525	179	279	171	81
Balanced $0.1 \leq d < 0.5$	367	41	78	119	0
Dense $0.5 \leq d$	182	12	50	41	0
Total (2,125)	1,074	232	407	331	81
Benchmark	DAv2	DAv2	DAv2 and DAv3	DAv2 and DAv3	DAv3

Table 2: MQLib instance counts across categories defined by size and density. The total number of our chosen instances is 2,125. Small and x-Small categories are analyzed in Appendix A for DAv2.

DA vs. best MQLib heuristics

The full MQLib [21] includes instances from the following sources: Gset [22], ORLib [65], SteinLib [66], 2nd, 7th, and 11th DIMACS [67–69], TSPLib [70], Biq Mac [71], and more from original solver papers. Furthermore, it includes random problem instances generated from multiple random generators: Culberson [72], NetworkX [73], and Rudy [74]. At the time of writing this paper, MQLib contains a total number of 3506 instances. Exactly 3,396 instances were introduced by the original MQLib paper [17], while the remaining instances were later added by external contributors. The instances span a wide range in both size n (number of vertices), from as few as 3 to approximately 53,000 variables, and density d (the ratio of actual to possible edges), ranging from extremely sparse graphs with $d \sim 0.0001$ to fully connected graphs with $d = 1$.

From the full instance set, we consider only instances with baseline time limits above 0.25 seconds. This reduces the total number of instances in MQLib to 2,125. With this, we filter out particularly easy instances, as well as those for which the solvers are likely to far exceed the assigned time limit due to a particularly short baseline time. Most solvers do not check the time limit continuously; instead, they check it after, e.g. , a certain number of internal iterations, or after completing a certain operation. If the time limit is relatively short, the solver may overshoot it significantly before reaching the next checkpoint.

We divide our selected instances into 15 categories based on size and density. The exact number of instances that fall in each of the size-density category combinations is shown in Table 2. In Section 4.1, we compare DAv2 on Medium–Large instances (up to what fits the DAU) and DAv3 on Medium–x-Large instances, across all density categories, against the best-performing MQLib heuristic on each category. In these selected categories, the solvers are more likely to adhere to the instance-specific time limits, which helps to ensure the justifiability of the comparison. The categories involved in the main analysis are visualized in Fig. 1 by the regions bounded by dashed lines. Most instances of this selection lie in the Medium- and Large-size categories, with a higher concentration in the Sparse category. The MQLib appears to lack x-Large instances that are Balanced or Dense, which explains the empty regions in the middle and upper right of the scatter plot in Fig. 1.

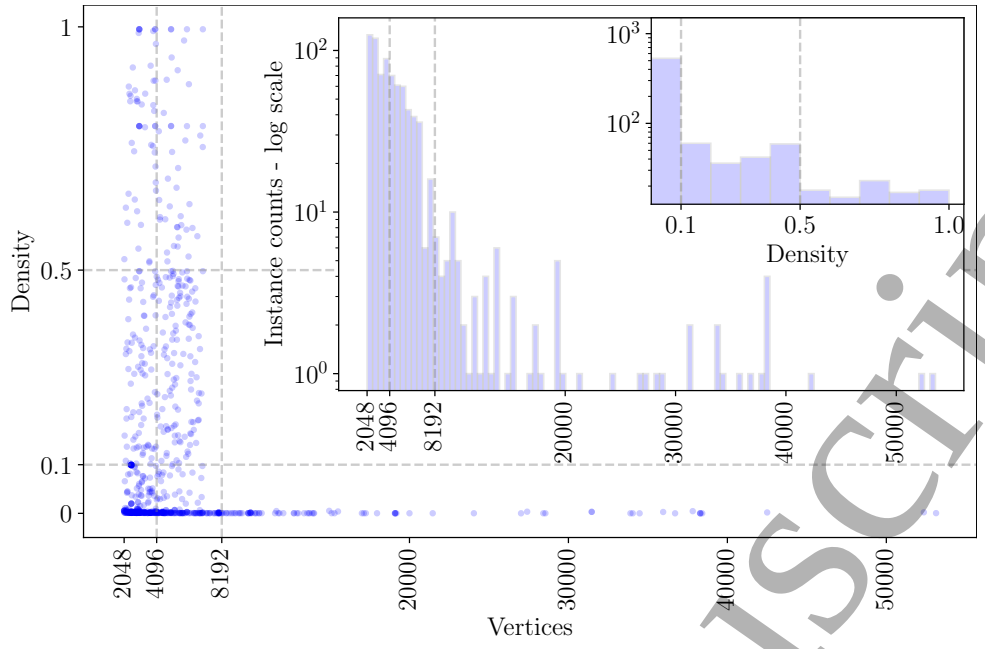


Figure 1: Distribution of instances by number of vertices and density. All instances above $\sim 8,000$ variables are sparse, whereas those with fewer variables exhibit a wider range of densities. Darker blue regions indicate a higher concentration of instances, which can be observed at low density and number of vertices. Inset: Histogram (log scale) showing instance counts by size range; the majority of instances are below $\sim 8,000$ variables. Nested inset: Histogram (log scale) showing instance counts by density range; the majority of instances are sparse.

DAv3 vs. D-Wave's hybrid solver

Next, we follow D-Wave's selection of instances from the MQLib based on three criteria [19]. First, they consider connected instances only with size between $n = 1,000$ and $n = 10,000$. Second, all selected instances were reported to have a maximum recommended time limit of 20 minutes. Third, the best objective value of each instance should be achieved by at most two heuristics out of the 37 MQLib heuristics. These criteria yield a subset of 45 instances ranging from $\sim 2,000$ to 10,000 in size and ~ 0.0004 to ~ 0.97 in density [19]. We noticed that 39 instances were assigned a time limit below 20 minutes in the original study, and only 6 were assigned a time limit above 20 minutes. Moreover, we observed that three instances in this subset have more than two heuristics achieving the best objective value. However, we proceed with D-Wave's selection [19] and make a comparison of DAv3 with D-Wave's HS.

DA vs. QIS3

The comparison between DAv2, DAv3, and the recently introduced quantum-inspired solver QIS3 by Yang *et al.* [20] is conducted on a subset of 16 instances from the G-set [22], with graph sizes ranging from 800 to 10,000 variables that are very sparse (~ 0.0004 to 0.01). The G-set comprises a total of 71 instances,

Density \ Size	x-Small	Small	Medium	Large	x-Large
Sparse	▲ 205 (39.0%)	▲ 85 (47.5%)	▲ 111 (39.8 %)	▲ 56 (32.7%)	★ 37 (45.7%)
Balanced	♦ 183 (49.9%)	♦ 13 (31.7%)	♦ 39 (50.0%)	♦ 66 (55.5%)	-
Dense	♦ 119 (65.4%)	♦ 6 (50.0%)	♦ 27 (54.0%)	♦ 28 (68.3%)	-

▲ BURER2002 ♦ PAL2004bMTS2 ★ MERZ1999GLS

Table 3: Best-performing MQLib heuristics across instance categories, symbol-coded by heuristic. The number in each cell indicates the amount of instances the corresponding heuristic achieved the best objective value for, based on the original data by Dunning *et al.* [17].

but the rationale behind the selection of this particular subset is not discussed. Notably, two of the selected instances exceed the 8,192-variable capacity of a single DAU and therefore cannot be solved using DAv2.

3.3. Choosing the best-performing MQLib heuristics

Ref. [17] implements and compares 37 heuristics based on a variety of meta-heuristic approaches, including simulated annealing, cross-entropy methods, non-linear optimization, tabu search, global equilibrium search, greedy randomized adaptive search (GRASP), estimation of distribution algorithms, genetic algorithms, and some of their variants. In Section 3.1, we obtained a new baseline time limit per instance; we therefore aim to compare the DA with new runs of MQLib heuristics under these time limits. For this purpose, we select the best-performing heuristic within each size and density category defined in Section 3.2. We define the best heuristic per category as the one that achieves the best objective value for the highest number of instances in the corresponding category, based on the original data by Dunning *et al.* [17]. These heuristics were then executed on the same server used to establish the baseline time limits and to run the DA. Table 3 lists the selected MQLib heuristics for each category, along with the number of instances within that category for which each heuristic found the best solution. BURER2002 is a non-linear optimization method combined with local search [49]; PAL2004bMTS2 is an iterated tabu search [50]; and MERZ1999GLS is a genetic algorithm incorporating crossover and local search [75]. As shown in Table 3, BURER2002 performs best on Sparse instances while PAL2004bMTS2 is most effective on Balanced and Dense instances. MERZ1999GLS shows the best performance on x-Large Sparse instances. After selecting the best-performing heuristics, we rerun these heuristics in their default settings as provided in [21] on the relevant instances using the newly assigned time limits.

4. Results

In this section, we present the benchmark results for the main study comparing DAv2 and DAv3 with MQLib heuristics in Section 4.1, as well as the complementary comparisons of DAv3 with D-Wave's HS in Section 4.2, and

of both DAv2 and DAv3 with QIS3 in Section 4.3. DAv3 often exceeds short time limits; therefore, we restrict it to Medium to x-Large instance sizes with relatively longer assigned time limits, which it can to some extent reliably solve within the assigned time, ensuring a meaningful comparison.

For these studies, we executed DAv2, DAv3, and MQLib heuristics on each instance with the specified time limit using five different random seeds, and here we report the highest achieved objective value, along with the minimum runtime among the runs that reached this value (3b). In addition, whenever applicable, we provide statistical inference using paired run-level significance tests.

To support reproducibility and facilitate further analysis, the GitHub repository accompanying this work [23] provides raw data and summary CSV files for each solver considered, listing per-instance minimum, maximum, median, mean, and standard deviation of both the objective values and runtimes across the 5 independent runs.

DAv2 was executed in simulated annealing mode, whereas DAv3 was executed in its only available mode, parallel tempering. For all instances, minor preprocessing was applied. Specifically, the heaviest variable in an instance (the one with the largest linear coefficient) was fixed, meaning it was preassigned to one side of the cut. Additionally, disconnected variables (i.e., $q_{ij} = 0$ for $x_i, x_j \in V$) are automatically excluded by the DA, as they do not contribute to the cut. As a result, the resulting QUBO problem typically contains $|V| - 1$ variables, though in some cases it may involve slightly fewer due to the removal of disconnected variables.

4.1. DA vs. best MQLib heuristics on Medium–x-Large instances

In Fig. 2, we compare DAv2 against the best-performing MQLib heuristic per category (as selected in Section 3.3) using a cumulative bar plot. Each bar is divided into three segments: proportion of instances where DAv2 finds a better cut value than the selected category heuristic (green), proportion of instances where both solvers find the same cut value (yellow), and proportion of instances where the heuristic finds a better cut value than the DAv2 (blue). The results show that DAv2 yields higher cut values than the selected MQLib heuristics on the majority of instances across all categories, with its largest advantage on Sparse instances–achieving greater cut values than BURER2002 on over 75% of the instances.

Fig. 3 presents a bar plot of instance counts across cut ratio ranges, defined as the cut value achieved by DAv2 to that achieved by the corresponding heuristic. The plot shows a clear skew to values greater than 1.0 (i.e., higher green bars), highlighting an overall better result of DAv2 than the best-performing category heuristics. Nearly all instances with cut ratio below 0.998 are float-type instances. The DA natively operates on integer-valued QUBO coefficients. Instances with floating-point values require scaling and rounding before processing. Therefore, when the range between the smallest

and largest QUBO coefficient values q_{ij} of an instance is too wide, scaling the

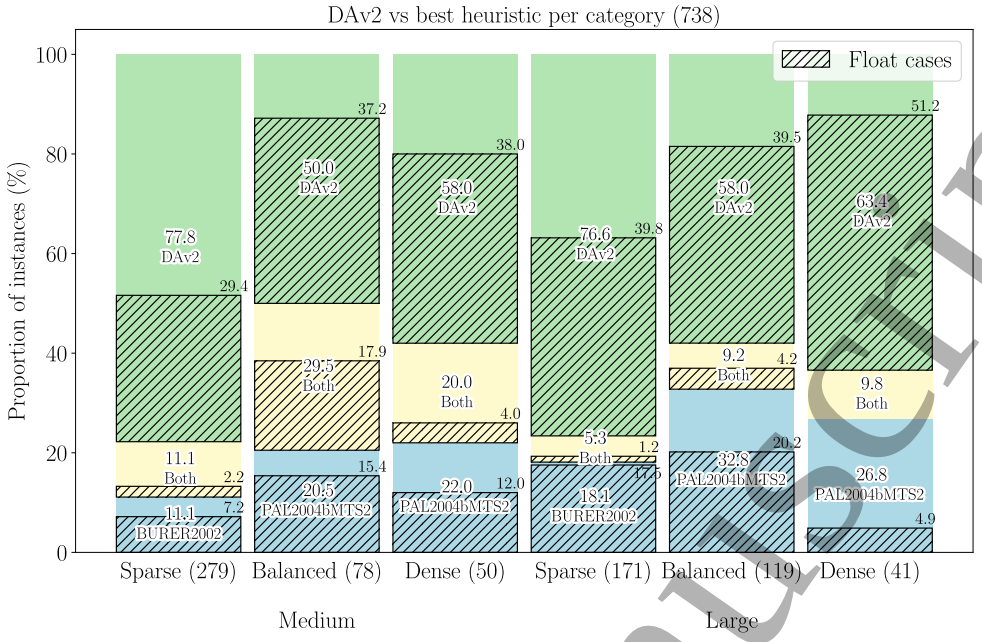


Figure 2: Comparison of the performance of DAv2 with the best-performing category heuristic on Medium and Large instances across all densities. Each bar represents the proportion of instances where DAv2 performs better (green), equal (yellow), or worse (blue) than the corresponding heuristic. Hashed bars indicate instances with floating-point cut values.

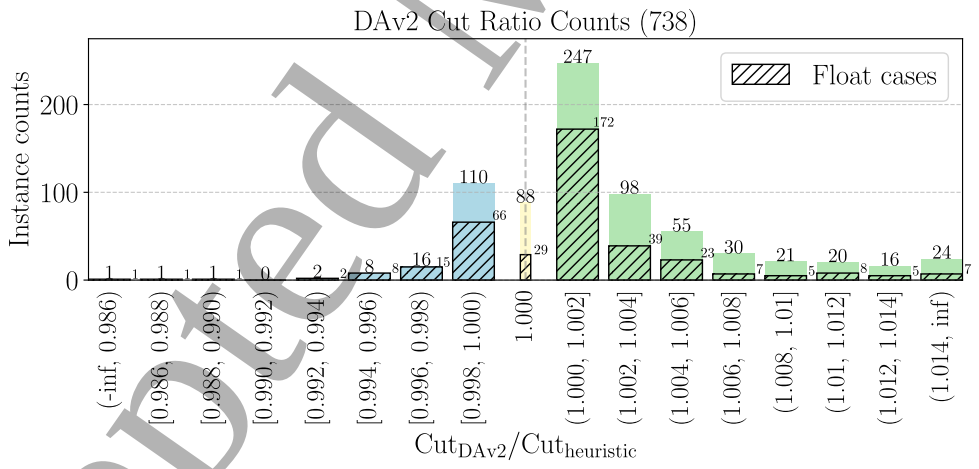


Figure 3: Distribution of Medium and Large instances over cut ratio ranges, where the cut ratio is defined as the DAv2 cut divided by that of the best-performing category heuristic. The 'tie' bar at 1.0 (yellow) shows the count of instances when both solvers found an equal cut value. The dashed line at 1.0 separates 'win' ranges (green bars) from 'loss' ranges (blue bars). Hashed bars indicate instances with floating-point cut values.

coefficients up by $2^{\alpha}/\max(q_{ij})$ with $\alpha = 63$ or $\alpha = 15$ for DAv2 (depending on the size of the QUBO matrix) and $\alpha = 48$ for DAv3 may still result in the smallest coefficients having values below 1. These are then rounded down to the nearest integer, and in this case, to zero, which results in the loss of important problem information and distorts the original QUBO problem.

In total, based on best-of-5-runs, DAv2 achieves better cuts in 511 and equal in 88 out of 738 instances, resulting in a 'win' rate of $\sim 69.24\%$, a 'tie' rate of $\sim 11.92\%$, and a 'loss' rate of $\sim 18.83\%$. To assess statistical significance we consider paired (by seed) run-level comparisons. DAv2 achieves better objective values than the corresponding category heuristics in a majority (≥ 3) of five paired runs on 542 of 738 instances ($\sim 73.44\%$), which is highly significant according to a two-sided paired sign (binomial) test ($p = 2.35 \times 10^{-38}$). Runs in which DAv2 and the corresponding category heuristics achieve identical objective values are conservatively counted in favor of the latter. This strict criterion constitutes a worst-case assumption for DAv2's performance and reduces the risk of overstating its advantage. The significantly low p-value shows that the observed dominance of DAv2 cannot plausibly be attributed to random variation.

We now turn to DAv3 and evaluate its performance on the Medium to x-Large instance categories using the same methodology. As shown in Fig. 4, DAv3 clearly achieves higher quality results than BURER2002 on Sparse instances, while PAL2004bMTS2 surpasses DAv3 on Large instances that are Balanced and Dense. The majority of 'wins' by the selected heuristics correspond to float-type instances, as indicated by the hashed bars–possibly due to the effect of rounding errors on the DA's cut values. In fact, the rounding error is upper bounded as $|x^{\top}Qx - x^{\top}\tilde{Q}x| \leq 2^{-\alpha}n^2 \max(q_{ij})$, where $\alpha = 48$ for DAv3 and $\tilde{Q}$ is the scaled and rounded QUBO matrix with coefficients $\tilde{q}_{ij} = \frac{\lfloor q_{ij} * 2^{\alpha}/\max(q_{ij}) \rfloor}{2^{\alpha}/\max(q_{ij})}$. This error bound explains the sensitivity of the DA to rounding errors with increasing instance sizes. Additionally, MERZ1999GLS shows a notable advantage over DAv3 on the x-Large Sparse category.

Compared to DAv2, both versions perform best on Medium and Large Sparse instances, with DAv3 showing a slight improvement. However, DAv2 achieves 26.9% and 29.3% improved results on Large Balanced and Large Dense instances, respectively. Similar to DAv2, the cut ratio bar plot for DAv3 in Fig. 5 shows a right-skewed distribution, indicating overall improved results compared to the best-performing category heuristics. However, unlike DAv2, most DAv3's 'losses' on float cases fall within $[0.998, 1)$ cut ratio range, suggesting a greater sensitivity to rounding errors. In total, DAv3 achieves better cuts in 498 and equal in 142 out of 819 instances, resulting in a 'win' rate of $\sim 60.81\%$, 'tie' rate of $\sim 17.34\%$, and a 'loss' rate of $\sim 21.85\%$. Using the same paired run-level comparison procedure described above, DAv3 achieves better objective values in a majority (≥ 3) of five paired runs on 517 of 819 instances ($\sim 63.13\%$), which is again highly significant according to a two-sided paired sign (binomial) test ($p \sim 5.45 \times 10^{-14}$).

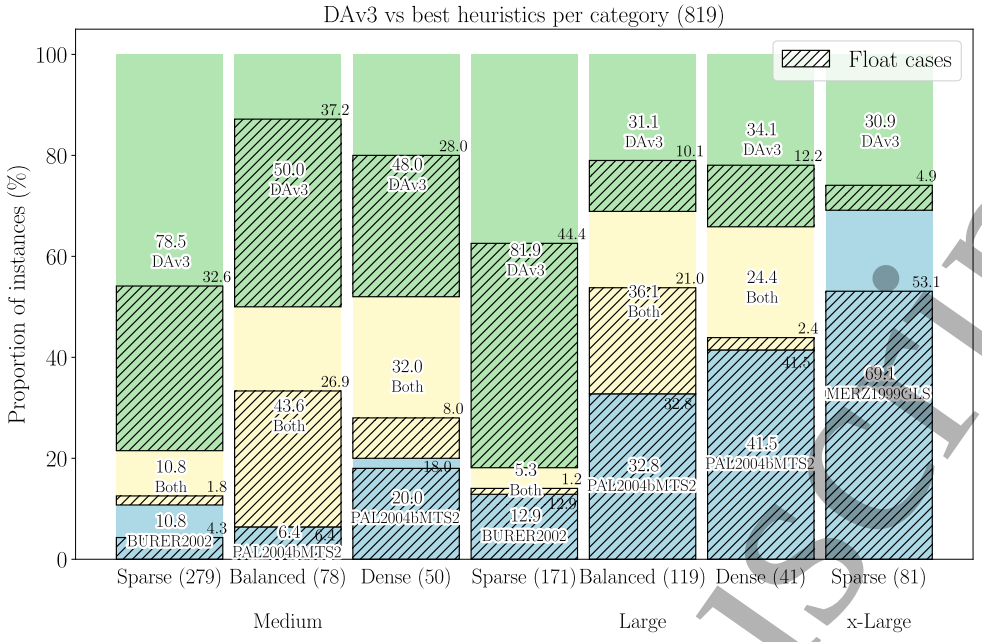


Figure 4: Comparison of the performance of DAv3 against the best-performing category heuristic on Medium, Large, and x-Large instances across all available densities. Each bar represents the proportion of instances where DAv3 performs better (green), equal (yellow), or worse (blue) than the corresponding heuristic. Hashed bars indicate instances with floating-point cut values.

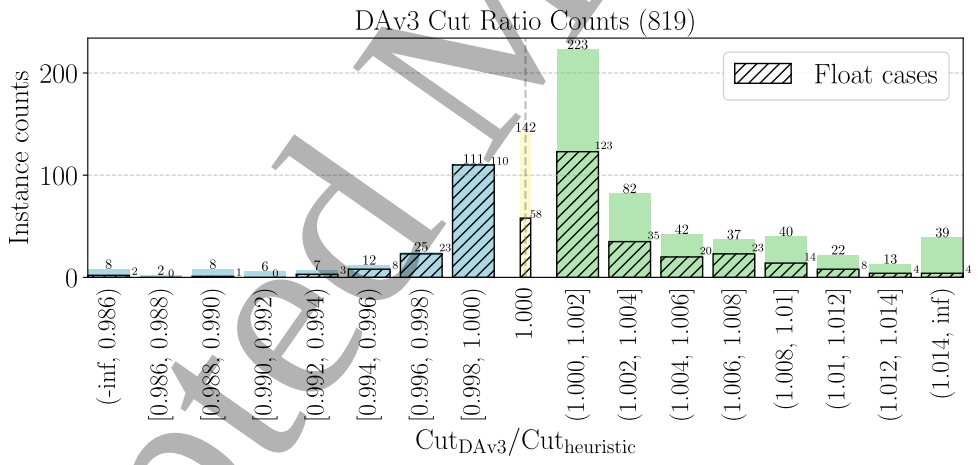


Figure 5: Distribution of Medium, Large, and x-Large instances over cut ratio ranges, where the cut ratio is defined as the DAv3 cut divided by that of the best-performing category heuristic. The 'tie' bar at 1.0 (yellow) shows the count of instances when both solvers found an equal cut value. The dashed line at 1.0 separates 'win' ranges (green bars) from 'loss' ranges (blue bars). Hashed bars indicate instances with floating-point cut values.

One advantage of partitioning the instance set by size is that it revealed a significant, initially unexplained degradation in the solution quality of DAv3 on the instances in the Large category. Upon investigation, we found that the internal automatic scaling of DAv3 is not efficient. To address this, we applied manual scaling. Our observations indicate that with 48-bit precision and instance sizes larger than 4,096 vertices, the solver partitions the problem into 4,096-variable chunks, which leads to degraded performance. By manually setting the precision to 64 bits[2] for $n \leq 4,096$ and 16 bits for $n > 4,096$ (mimicking the behavior of DAv2), instances larger than 4,096 vertices (still smaller than 8,192) can be embedded as a single chunk and solved in full, resulting in better performance as shown in Fig. 6. Thus, we provide an algorithmic rule (Algorithm 1 below) for coefficient scaling and a recommendation on when to trust the automatic scaling and when to choose the manual scaling.

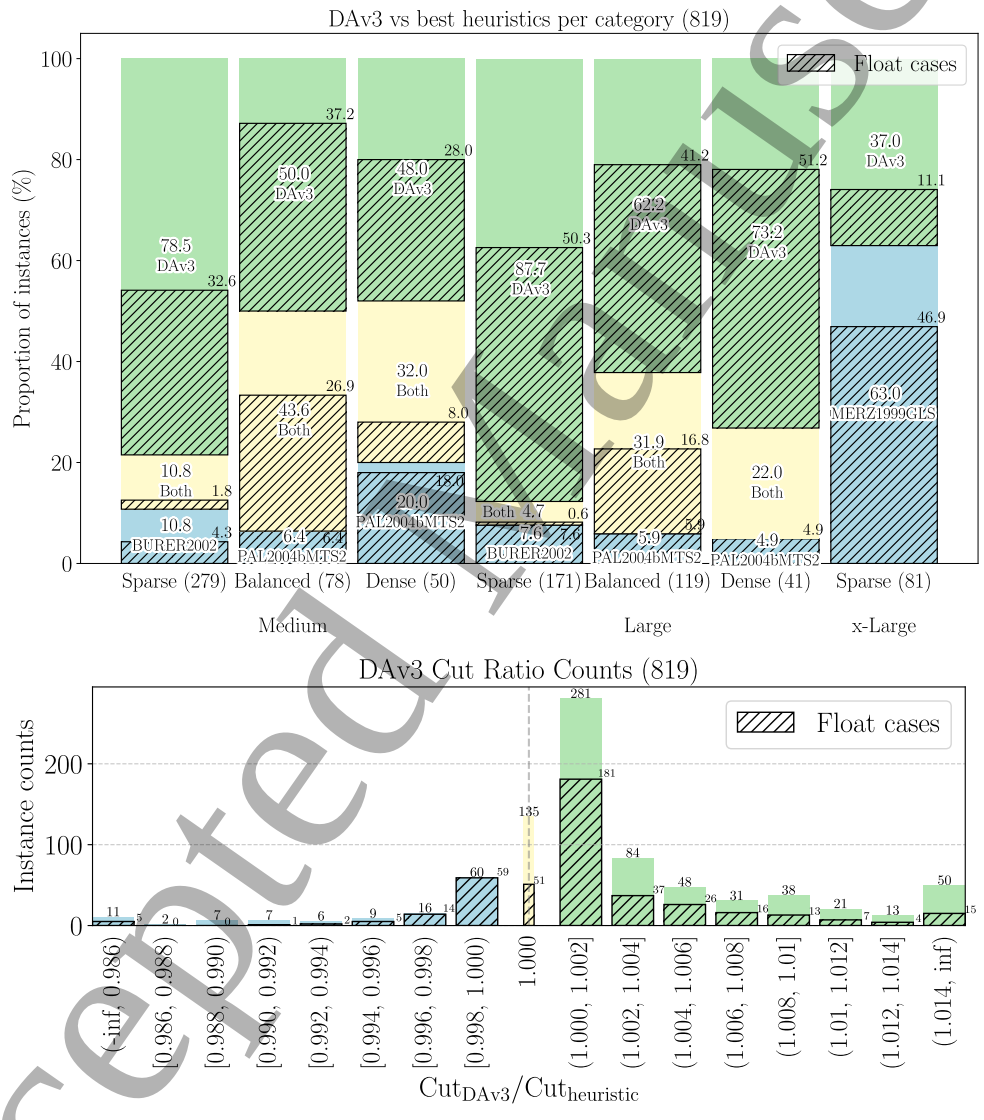


Figure 6: Improved results of DAv3 by the manual scaling described in Algorithm 1 instead of the internal automatic scaling.

[2]Later scaled down internally by DAv3 to 48-bit precision.

Algorithm 1: Selection rule for DAv3 scaling setting

Input: Instance size n and coefficient-type (int-only / float)
Output: Scaling setting for DAv3

```
if n ≤ 4,096 then
    if int-only then
        Use integer representation without scaling.
                                    // Scaling not needed
                            // All coefficients fit into int64
    else if float then
        Scale coefficients up so that they fit into int64 OR use DAv3
         automatic scaling.
          // Automatic scaling can be trusted in this regime
else if n > 4,096 then
    if int-only then
        if max coefficient fits into int16 then
            Use integer representation without scaling.
                                    // Scaling not needed
                            // All coefficients fit into int16
        else
            Scale coefficients down so that they fit into int16.
                                // Automatic scaling can be outperformed
    else if float then
        Scale (up or down) coefficients so that they fit into int16.
                                // Automatic scaling can be outperformed
Run DAv3 with the selected scaling setting.
```

The results confirm a substantial improvement for the Large category across all densities: the proportion of instances where DAv3 found a better cut increased by 5.8% for Sparse instances, 31.1% for Balanced instances, and 39.1% for Dense instances. A slight gain (6.1%) was also observed for x-Large Sparse instances, though larger improvements are not expected due to the unavoidable partitioning of the problem into 8,192-variable chunks. Overall, with this adjustment, DAv3 achieved better cuts in 566 and equal in 135 out of 819, resulting in a 'win' rate of $\sim$69.11%, 'tie' rate of $\sim$ 16.48%, and a 'loss' rate of $\sim$ 14.41%. Using the same paired run-level comparison procedure as before, DAv3 achieves better objective values in a majority (≥ 3) of five paired runs on 582 of 819 instances ($\sim$ 71.06%), which is again highly significant according to a two-sided paired sign (binomial) test ($p \sim 2.81 \times 10^{-34}$).

Table 4 presents the changes in the cut ratios (i.e. $\text{Cut}_{\text{DAv3}}/\text{Cut}_{\text{heuristic}}$) in greater detail before and after applying our manual scaling (Algorithm 1) in place of the automatic scaling used by DAv3. The diagonal entries of the table correspond to instance counts that remain within the same cut ratio range. Counts below the diagonal indicate improvements in the cut ratio, whereas counts above the diagonal indicate degradations. As shown in the table, the majority of changed instance counts lie below the diagonal, demonstrating improved performance under manual scaling. This finding highlights a trade-off between integer precision and problem decomposition, with greater gains

from / to	(-inf, 0.986)	[0.986, 0.988)	[0.988, 0.990)	[0.990, 0.992)	[0.992, 0.994)	[0.994, 0.996)	[0.996, 0.998)	[0.998, 1.000)	1.000	(1.000, 1.002]	(1.002, 1.004]	(1.004, 1.006]	(1.006, 1.008]	(1.008, 1.01]	(1.01, 1.012]	(1.012, 1.014]	(1.014, inf)	Total (after)
(-inf, 0.986)	8							2	1									11
[0.986, 0.988)		2																2
[0.988, 0.990)			7															7
[0.990, 0.992)			1	6														7
[0.992, 0.994)					5			1										6
[0.994, 0.996)					2	5	2											9
[0.996, 0.998)						4	12											16
[0.998, 1.000)							4	48	6	2								60
1.000									135									135
(1.000, 1.002]							3	58		220								281
(1.002, 1.004]						1	3	1		1	78							84
(1.004, 1.006]						2	1	1			4	39	1					48
(1.006, 1.008]												2	28	1				31
(1.008, 1.01]													5	33				38
(1.01, 1.012]														5	16			21
(1.012, 1.014]													1		2	10		13
(1.014, inf)												1	2	1	4	3	39	50
Total (before)	8	2	8	6	7	12	25	111	142	223	82	42	37	40	22	13	39	819

Table 4: Changes in cut-ratio distributions, shown as counts per cut-ratio range, before and after replacing the automatic scaling of DAv3 with the manual scaling rule in Algorithm 1. Diagonal entries indicate unchanged cut ratios (remaining in the same range), entries below (above) the diagonal indicate improvements (degradations).

achieved by solving the problem as a whole at lower integer precision.

Nevertheless, for consistency, we continue the following comparisons with D-Wave's HS and QIS3 using the default automatic scaling of DAv3.

4.2. DAv3 vs. D-Wave's hybrid solver

We benchmark DAv3 against D-Wave's HS [19] using their specified instance selection criteria, as also detailed in Section 3.2. The test set is further divided into integer (14 instances) and float (31 instances) cases. Figure 7 shows the progress of DAv3 solution quality relative to D-Wave's HS's solution, over a 20-minute time limit (left), alongside a summary cut ratio bar plot (right). The upper plots correspond to integer cases, and the lower plots to float cases. Each line corresponds to a given instance (best-of-5 runs), with crosses marking the time the final reported solution was found by DAv3.

For integer cases, DAv3 outperforms D-Wave HS with 8 'wins', 5 'ties', and 1 'loss'. The progress inset shows that for most instances, DAv3 finds an equal or better solution within the first few seconds. Under the tie-handling scheme in which paired runs yielding identical objective values contribute to 0.5 to each solver, an instance is counted as a win for DAv3 if its aggregated run-level score exceeds 2.5. According to this criterion, DAv3 outperforms D-Wave's HS on 9 of 14 integer instances (~ 64.29 %). However, this difference is not statistically significant according to a two-sided paired sign (binomial) test ($p \sim 0.424$), indicating that the observed 'win' imbalance is compatible with random variation.

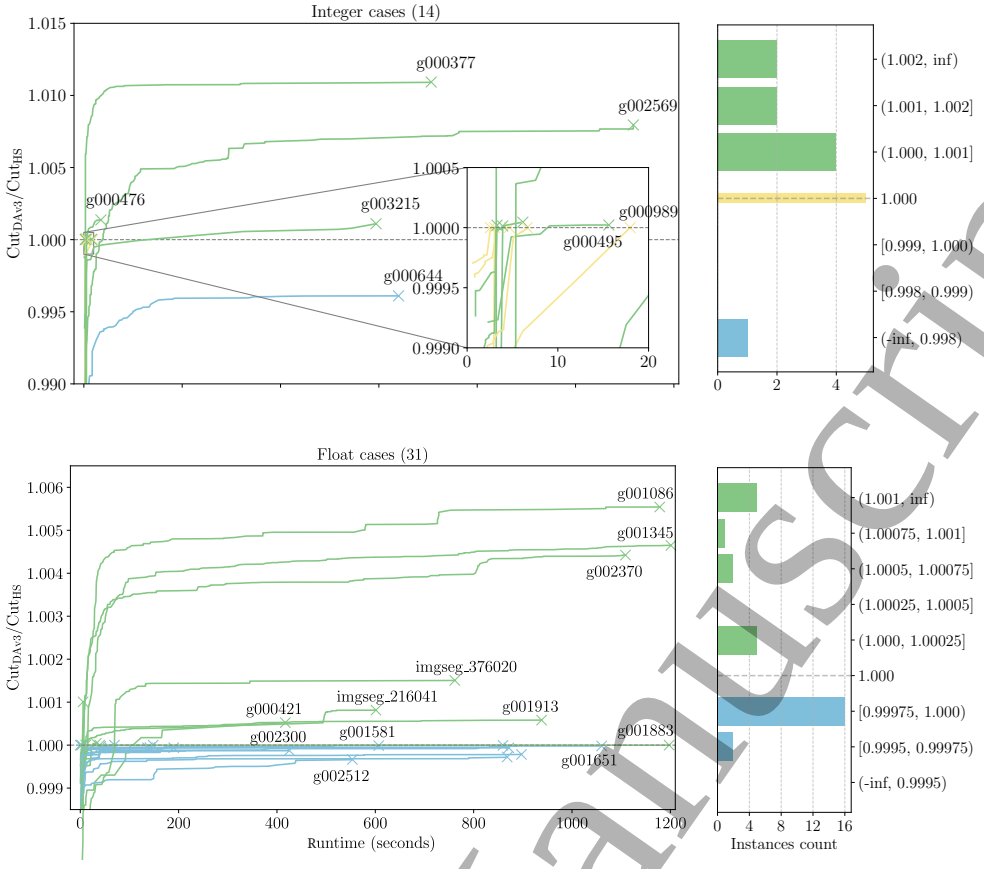


Figure 7: Left: Progress of DAv3 cut ratio relative to D-Wave's HS over the 20-minute time limit, separated by integer (top) and float (bottom) instance sets. Each line corresponds to an individual instance, with crosses marking the time when the best cut was found. Colors indicate the performance of DAv3 relative to D-Wave's HS: green for better, yellow for equal, and blue for worse. Right: Bar plot summarizing final cut ratio distribution.

For float instances, DAv3 underperforms D-Wave's HS on 18 instances and outperforms on 13 instances. Most 'losses' cluster near a cut ratio range of 0.99975 to 1. This issue is expected given the integer nature of the machine, i.e., encoding the coefficient of a binary quadratic polynomial as an integer. Notably, for many float instances, DAv3 finds better solutions in less than 20 minutes, with 5 instances in just a few seconds. DAv3 achieves better instance-level results on 10 of 31 float instances ($\sim 32.26\,\%$). The corresponding two-sided paired sign (binomial) test does not indicate a statistically significant difference ($p \sim 0.071$), suggesting that performance differences remain inconclusive. Applying the manual scaling discussed in the previous section improves the results of DAv3 on the float-instances, reducing the number of 'losses' from 18 to 13.

The rapid convergence of DAv3 DAv3 (3a) to its best solution, often matching or surpassing those of D-Wave's HS, highlights the practical potential of for runtime-sensitive scenarios.

			QIS3	DAv2		DAv3		
name	n	d	cut	cut	time (sec)	cut	time (sec)	limit (sec)
G11	800	0.005	**564**	**564**	9.866	**564**	10.983	10
G32	2000	0.002	1404	**1410**	9.91	**1410**	9.652	8
G48	3000	0.0013	**6000**	**6000**	9.838	**6000**	8.795	8
G57	5000	0.0008	3466	3470	9.959	**3482**	8.737	6
G62	7000	0.0006	4828	4838	9.961	**4846**	10.759	7
G65	8000	0.0005	5502	5460	10.139	**5534**	12.166	8
G66	9000	0.0004	**6288**	-	-	6180	8.772	8
G72	10000	0.0004	**6916**	-	-	6728	9.029	4
G14	800	0.0147	3060	**3064**	9.866	**3064**	10.978	10
G51	1000	0.0118	3846	**3848**	9.869	**3848**	11.034	9
G35	2000	0.0059	7673	**7686**	9.903	**7686**	9.635	8
G58	5000	0.0024	19216	19262	9.956	**19263**	8.893	8
G63	7000	0.0017	26949	**27012**	9.942	27003	10.873	8
G1	800	0.06	**11624**	**11624**	9.864	**11624**	10.995	10
G43	1000	0.02	**6660**	**6660**	9.871	**6660**	11.041	9
G22	2000	0.01	13358	**13359**	9.901	**13359**	9.725	9

Table 5: Comparison of DAv2, DAv3, and QIS3 on G-set instances [22]. The results for QIS3 are taken from Yang *et al.* [20], while the DA solvers were benchmarked under similar 10-second runtime constraints. The actual runtime for DAv2 and DAv3 is reported for each instance. G66 and G72 exceed the 8,192-variable capacity of DAv2 and thus yield no results for that version.

4.3. DA vs. QIS3

During the preparation of this manuscript, a new metaheuristic QUBO solver, QIS3, was introduced by Yang *et al.* [20]. QIS3 combines branch-and-bound pruning with gradient-descent refinement for intensified local exploration, and a quantum annealing-inspired bounding technique to accelerate the pruning. It also incorporates adaptive strategies to optimize performance across diverse problem instances. In their work, the authors benchmark QIS3 on a subset of 16 instances from the G-set collection [22], against eight state-of-the-art solvers: their previous version QIS2, a genetic algorithm, the coherent Ising machine, simulated annealing, parallel tempering, simulated bifurcation, D-Wave's simulated annealing (Neal), and Gurobi. All algorithms were evaluated under 10-second runtime constraints. Therefore, we configure the parameters of DAv2 as described in Appendix D to achieve a runtime of approximately 10 seconds. For DAv3, we run the solver with several time offsets ranging from 0 to 9 seconds and select the offset that yields a runtime closest to 10 seconds. In their original study, QIS3 reported the highest cut value on 15 out of 16 instances (with the exception of instance G58).

Results of DAv2, DAv3, and QIS3 (as reported in Ref. [20]) are presented in Table 5. For the DA, both the time limit and actual runtimes are reported (2b), while only the time limit is available for QIS3. Notably, one or both versions of the DA find better or matching results than QIS3–and thus than all eight solvers in Ref. [20]– on 14 out of the 16 instances, including G58. The only exceptions are instances G66 and G72, where QIS3 remains the top

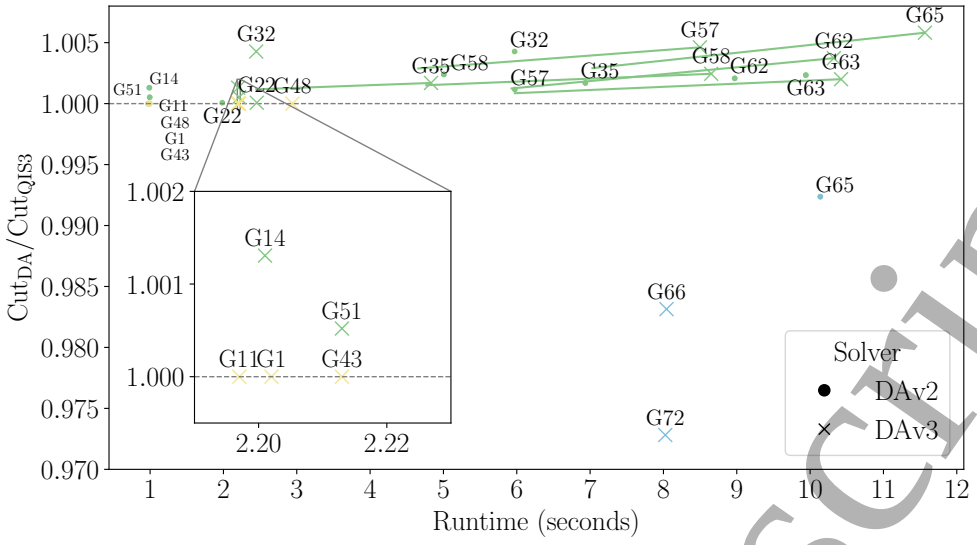


Figure 8: Progress of DAv3 solution quality relative to QIS3 over the 10 seconds time limit. Each line corresponds to an individual instance, with crosses marking the time when the best solution was found. Each dot corresponds to the cut quality of DAv2 on a given instance, relative to QIS3, measured at the earliest time the best cut was found across independent runs ranging from 1 to 10 seconds. Colors indicate the performance of the DA relative to QIS3: green for better, yellow for equal, and blue for worse. A progress plot for DAv2 is not possible due to the lack of the functionality to log intermediate improving cuts. The dots of instances G11, G48, G1, and G43 for DAv2 are overlapping.

performer, and G65, where only DAv3 found a better cut than QIS3. The drop in performance for DAv3 on G66 and G72 is attributed to the required partitioning when the problem size exceeds the 8,192-variable capacity of a single DAU.

Comparing both DA versions across the 14 instances where results are available for both, DAv3 yields better results more frequently than DAv2. DAv2 finds a larger cut than DAv3 only on G63, while it fails against both DAv3 and QIS3 on G65.

Figure 8 shows the progress of DAv3's cut value relative to QIS3's over the runtime. For half of the instances, DAv3 achieves an equivalent or better cut within the first few seconds. An equivalent figure for DAv2 is not possible because it lacks the functionality to log intermediate improving cuts into a solution pool during execution. However, we also display in Fig. 8 the best cut achieved by DAv2 (dots), as reported in Table 5, along with the shortest runtime at which it was attained. These results are derived from multiple runs with time limits ranging from 1 to 10 seconds. For 6 out of 14 instances, DAv2 found its best cut that matches or exceeds QIS3's cut in just a few seconds. For 4 instances, the cut was achieved in roughly half the time limit. This demonstrates the rapid convergence behavior of the DA toward high-quality solutions (3a).

5. Conclusion

We have conducted a comprehensive benchmarking study of Fujitsu's second- and third-generation Digital Annealers (DAs), DAv2 and DAv3, on the Max-Cut problem. We compare their performance against selected best-performing MQLib classical heuristics, D-Wave's hybrid quantum-classical solver (HS), and a recently introduced quantum-inspired heuristic QIS3.

Our evaluation across 2,125 instances–spanning graph sizes from 200 up to approximately 53,000 variables–was based on the best objective value achieved within instance-specific time limits, ensuring as fair a comparison as possible. The results demonstrate that DAv2 consistently found a better cut than the best classical heuristics on approximately 69.2% of the benchmarked subset of instances, while DAv3 maintains competitive performance with a success rate of about 60.8%. Notably, DAv3 internal automatic scaling and rounding to integer coefficients proved suboptimal. By applying manual scaling, however, the success rate improved to approximately 69.1%. On 45 Max-Cut instances selected by D-Wave, DAv3 outperformed D-Wave's HS, particularly on instances with integer weights. DAv3 found lower-quality cuts for the majority of float-valued instances, though with a high relative cut ratio. Finally, our comparison with QIS3 on 16 G-set instances reveals that one or both DA versions achieve shorter runtimes and better solutions in most cases. However, for two large G-set instances, QIS3 shows a surprisingly good performance. We further show that the DA can have a rapid convergence behavior already before the time limits are met. Often, high-quality solutions can be reached within the first few seconds of runtime.

Future work can expand this study in several directions. First, we aim to benchmark the DA on constrained NP-hard problems to assess its effectiveness in more complex settings. Our results also suggested that performance is highly instance-dependent: for certain graph structures or sizes, the DA quickly finds optimal or near-optimal cuts, while in other cases, traditional solvers or heuristics outperform it. This variation motivates a hybrid workflow that leverages different solvers depending on the instance features. Another promising direction is to incorporate preprocessing techniques into the DA, tailored to a specific problem class such as the Max-Cut, where it could further enhance its performance.

We emphasize that future benchmarking efforts would benefit from including modern parallelized or GPU-based classical solvers, such as multithreaded simulated annealing, parallel tempering, or GPU-accelerated heuristics. While our study is limited by the absence of such solvers, due to lack of publicly available and user-friendly implementations, we view their integration as a natural and important next step. We therefore recommend future works on such solvers to adopt and extend on our benchmarking design to complement and build upon our results presented here.

6. Conflict of interest

The DA is a Fujitsu product, and Kirsch, Münch, Schinkel, and Walter work for the *Fujitsu Germany GmbH*, which is a German Fujitsu subcompany. Their main work consists of consultations on quantum-inspired computing. Kliesch is the holder of the endowed professorship "Quantum Inspired and Quantum Optimization", which is financed by Fujitsu Services GmbH and the Dataport AöR. By German laws, all university professorships are guaranteed scientific independence, and this requirement is also accommodated in the legal framework of this endowed professorship.

The authors declare that the comparison of the DA with other methods was conducted to provide a fair and objective comparison, without giving special consideration to Fujitsu's interests. In fact, Winkler, Shaglel, and Kliesch initiated this work to independently determine the DA's performance without having to trust other sources. All authors made their best efforts to follow fair benchmarking standards as much as possible, given the technical feasibility.

7. Acknowledgment

We thank John Silberholz for helpful discussions about methodology of Ref. [17] and valuable comments regarding the MQLib repository [21]. We thank Mirko Arienzo, Nikolai Miklin, and Michel Krispin for discussions and valuable feedback during the development of this project. Shaglel and Kliesch are funded by Fujitsu Germany GmbH and Dataport as part of the endowed professorship "Quantum Inspired and Quantum Optimization" and the Hamburg Quantum Computing project, which is co-financed by the ERDF of the European Union and the Fonds of the Hamburg Ministry of Science, Research, Equalities and Districts (BWFGB).

Appendices

A. DAv2 vs. best MQLib heuristics on x-Small–Small instances

We analyze the performance of DAv2 in comparison with the corresponding best-performing MQLib heuristics for x-Small and Small categories across all densities. As identified in Table 3 in Section 3.3, the best heuristic for x-Small and Small Sparse instances is BURER2002, while for the Balanced and Dense instances it is PAL2004bMTS2. This analysis is omitted for DAv3 due to often exceeding the time limit specified for instances in these categories by significantly more than 10%, leading to an unfair comparison with other solvers (see Appendix F). Figure 9 illustrates the size and density characteristics of the instances involved in this analysis, with a total of 1,306 instances. Most

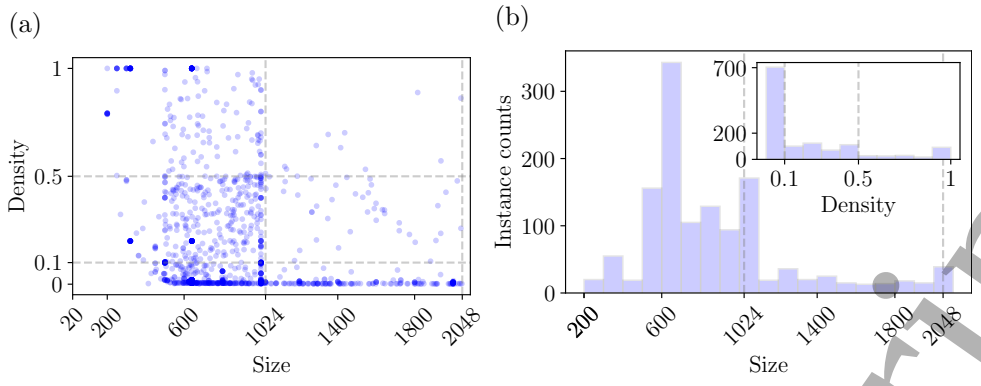


Figure 9: Distribution of instances by number of vertices and density for x-Small and Small category instances. The total number of instances is 1,306. (a) Scatter plot of instances in terms of their size and density. Darker blue regions indicate a higher concentration of instances at this size and density region. (b) Histogram showing instance counts by size range; the majority of instances are below $\sim 1,000$ variables (in the x-Small category). Inset: Histogram showing instance counts by density range; the majority of instances are sparse.

Balanced and Dense instances contain fewer than $\sim 1,000$ vertices, and the majority of x-Small and Small instances are sparse, as shown by the nested inset in Fig. 9b and the darker blue region of Fig. 9a.

Figure 10 shows that on average, DAv2 provides better solutions than BURER2002 in the corresponding categories: x-Small Sparse and Small Sparse (68.8% and 64.2%). PAL2004bMTS2 outperforms DAv2 on x-Small Dense (31.3% vs. 15.4%). For a considerable number of instances, especially in x-Small Balanced and Dense, both find the same cut value. On the other hand, DAv2 yields better results than PAL2004MTS2 on Small Balanced and Small Dense categories, finding a better cut in 53.7% and 58.3% of the instances, respectively. However, the number of instances in these categories is relatively small, so these results should be interpreted with caution.

Figure 11 presents a bar plot of instance counts across cut ratio ranges, defined as the ratio of the cut value achieved by DAv2 to that achieved by the corresponding heuristic. The plot shows a clear skew to the values greater than 1.0 (i.e., higher green bars), highlighting an overall better performance of DAv2 over the best-performing MQLib heuristics on the x-Small and Small categories. Most 'losses' of DAv2 have high cut ratio between $[0.998, 1.000)$. In total, DAv2 achieves better cuts in 612 instances and equal in 506 out of 1306 instances, resulting in a 'win' rate of $\sim 46.86\%$, a 'tie' rate of $\sim 38.74\%$, and a 'loss' rate of $\sim 14.40\%$. As explained in the main text, we consider paired (by seed) run-level comparisons to assess statistical significance. On these categories, DAv2 achieves better objective values in a majority (≥ 3) of five paired runs on 788 of 1306 instances ($\sim 60.34\%$), which is highly significant according to a two-sided paired sign (binomial) test ($p = 8.03 \times 10^{-14}$). Runs in which DAv2 and the corresponding category heuristic achieve identical objective values are conservatively counted in favor of the latter.

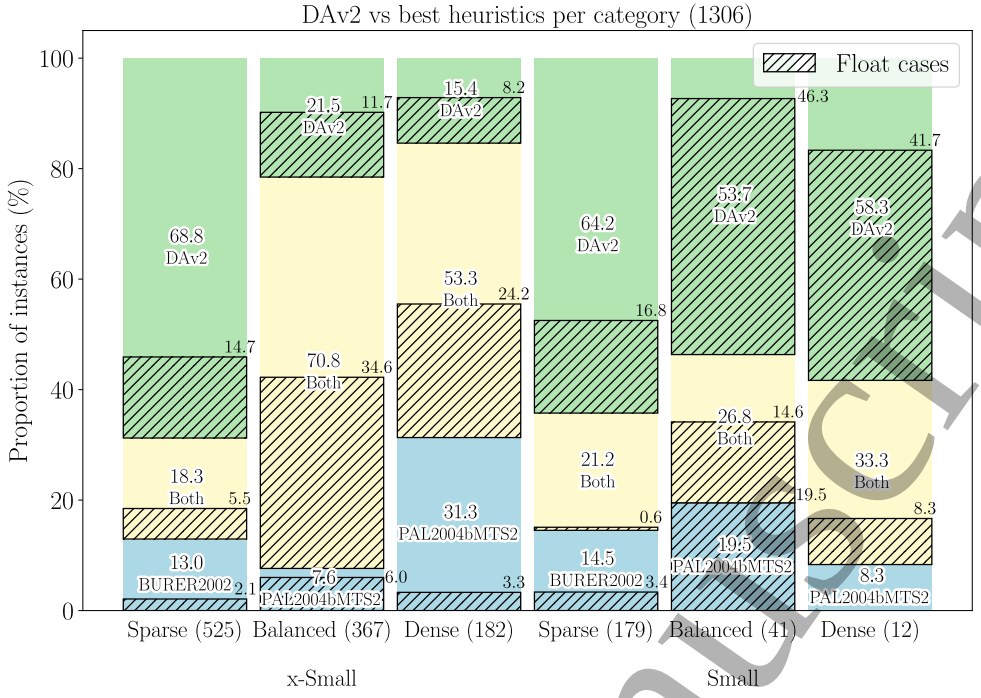


Figure 10: Comparison of the performance of DAv2 with the best-performing MQLib heuristic on x-Small and Small instances across all densities. Each bar represents the proportion of instances where DAv2 yields a better cut (green), equal (yellow), or worse (blue) than the corresponding heuristic. Hashed bars indicate instances with floating-point cut values.

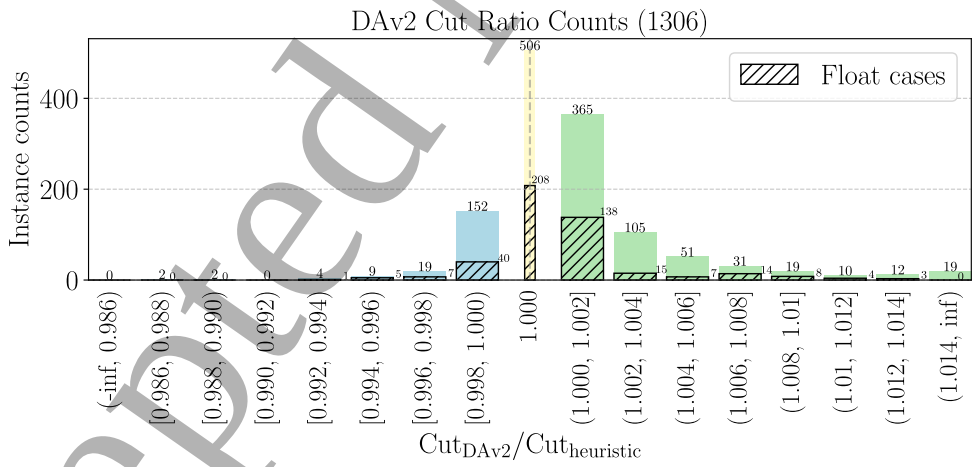


Figure 11: Distribution of all x-Small and Small instances over cut ratio ranges, where the ratio is defined as DAv2 cut divided by that of the best-performing category heuristic. The 'tie' bar at 1.0 (yellow) shows the count of instances when both solvers found an equal cut value. The dashed line at 1.0 separates 'win' ranges (green bars) from 'loss' ranges (blue bars). Hashed bars indicate instances with floating-point cut values.

B. Performance analysis on instances with known optimal solutions

The performance of heuristic solvers is typically tested on large Max-Cut instances, as such instances reveal scalability and robustness and better reflect real-world problem sizes, thereby enabling meaningful discrimination between strong and weak solvers. For large Max-Cut instances, the optimal solution is usually unknown, which is standard in the field [17, 20, 71, 76]. Consequently, solvers are commonly assessed using best-known cut values or through relative comparisons against established baseline heuristics, as done in this work.

To complement this evaluation, we additionally compare DAv2 and DAv3 against the best-performing heuristics identified in Section 3.3 on instances with known optimal solutions. Owing to the considerations outlined above, these instances are necessarily small.

We select multiple instance classes with known optimal solutions from BiqMac library [71], which are also included in MQLib [21], as listed in Table 6. Each instance class comprises multiple instances (indicated by the column number) and varies in size and density, as reflected by n and d columns. The exact specifications of the instances are provided in library's documentation [71].

For each solver, we report the number of instances per class for which the optimal solution is found within the instance-specific time limit (see Section 3.1), as well as the worst accuracy, i.e. achieved cut/optimal cut, achieved across all instances in that class.

For `bqp`, `pm`, `gk`, `g05`, and `pw` instance classes, nearly all solvers found the optimal solution for all instances. The main exception occur for BURER2002, which failed to reach optimality on one instance in `bqp` class (albeit with high accuracy), on five instances in `gk` class (with relatively low worst accuracy), and on two instances in `pw` class (with high worst accuracy). For `w` instance class, all solvers found the optimal solution for all instances except DAv2, which failed on a single instance but still achieved high accuracy.

The classes `ising` and `t` appear to be the most challenging. For `t` class, none of the solvers found the optimal solution for all instances; DAv3 performed best, finding the optimum for 13 instances. Similarly, for the `ising` class, no solver achieved optimality across all instances. Here, DAv3 again found the optimal solution for the largest number of instances, while DAv2 did not for any instance.

All instances in these classes have integer weights that fit within 64-bit signed integer, so numerical rounding errors are unlikely to play a role. The results suggest that, particularly for the `ising` instances, the assigned time limit underestimates the problem hardness. Recall that DAv3 often exceeds short imposed time limits (see Appendix E and Appendix F), which may explain its superior performance on some instance classes.

Instances	n	d	#	DAv2		DAv3		BURER2002		PAL2004MTS2		MERZ1999GLS	
				# opt	worst acc.	# opt	worst acc.	# opt	worst acc.	# opt	worst acc.	# opt	worst acc.
bqp	100	0.1	10	10	1	10	1	9	0.9973	10	1	10	1
ising	100,200,300	1	30	0	0.9918	6	0.9882	1	0.9869	3	0.9918	4	0.9946
w	100	0.1,0.5,0.9	30	29	0.9956	30	1	30	1	30	1	30	1
t	100–400	0.01025–0.04838	18	9	0.9883	13	0.9881	8	0.9901	2	0.9833	9	0.9819
pm	80,100	0.1,0.99	40	40	1	40	1	40	1	40	1	40	1
gk	20–100	0.0625–1	40	40	1	40	1	35	0.8836	40	1	40	1
g05	60,80,100	0.5	30	30	1	30	1	30	1	30	1	30	1
pw	100	0.1,0.5,0.9	30	30	1	30	1	28	0.9970	30	1	29	0.9975

Table 6: Solver Performance comparison on instances with known optimal solutions from BiqMac Library [71]. Worst accuracy achieved by the solver across all instances in a given class is reported.

C. Reliability of solvers measured by Coefficient of Variation (CV)

In the main text, we reported performance comparisons between the DA and the baseline solvers using the best objective value obtained over 5 independent runs with 5 different seeds per instance. This choice was motivated by an application-oriented setting in which practitioners seek the best solution obtained from a small number of independent trials. To further show that this choice does not bias the reported conclusions – and in addition to the statistical inference assessment that are independent of the best-of-5 metric – we explicitly analyze the run-to-run variability by presenting distribution of the coefficient of variation (CV) across the 5 runs objective values for each instance. The CV is defined as

$$\mathrm{CV} = \frac{\text{Standard deviation}}{\text{Mean}}, \tag{6}$$

which provides a scale-normalized and dimensionless measure of run-to-run variability. Fig. 12 shows these CV distributions as box plots grouped by instance category and solver. The upper plot corresponds to the study on Medium to x-Large instances presented in Section 4.1, while the lower plot corresponds to the x-Small to Small instances study discussed in Appendix A.

Overall, Fig. 12 shows that CV values are very small across all solvers and categories, indicating a high degree of robustness and limited stochastic variability in the obtained objective values. Across all applicable categories, DAv2 exhibits the lowest variability, as reflected by the lowest mean (green triangle) and median (orange horizontal line) CV values. DAv3 is generally more reliable than BURER2002 and PAL2004MTS2 heuristics, but less reliable than MERZ1999GLS. Notably, PAL2004MTS2 produces several extreme outliers for the Medium - Large and Balanced - Dense instances, which is reflected by the high CV mean value. We speculate that for larger and denser graphs the outcome concentrate more sharply around the typical behaviour. For too easy graphs, the behaviour also concentrates at the optimal solution.

Figure 13 further reports the distributions of CV of the runtime across the 5 runs per instance. The MQLib category heuristics generally exhibit lower run-to-run runtime variability than both DA v2 and v3 across almost all instance categories. DAv2 is clearly more reliable than DAv3. Nevertheless, the overall CV values remain small for all solvers and categories.

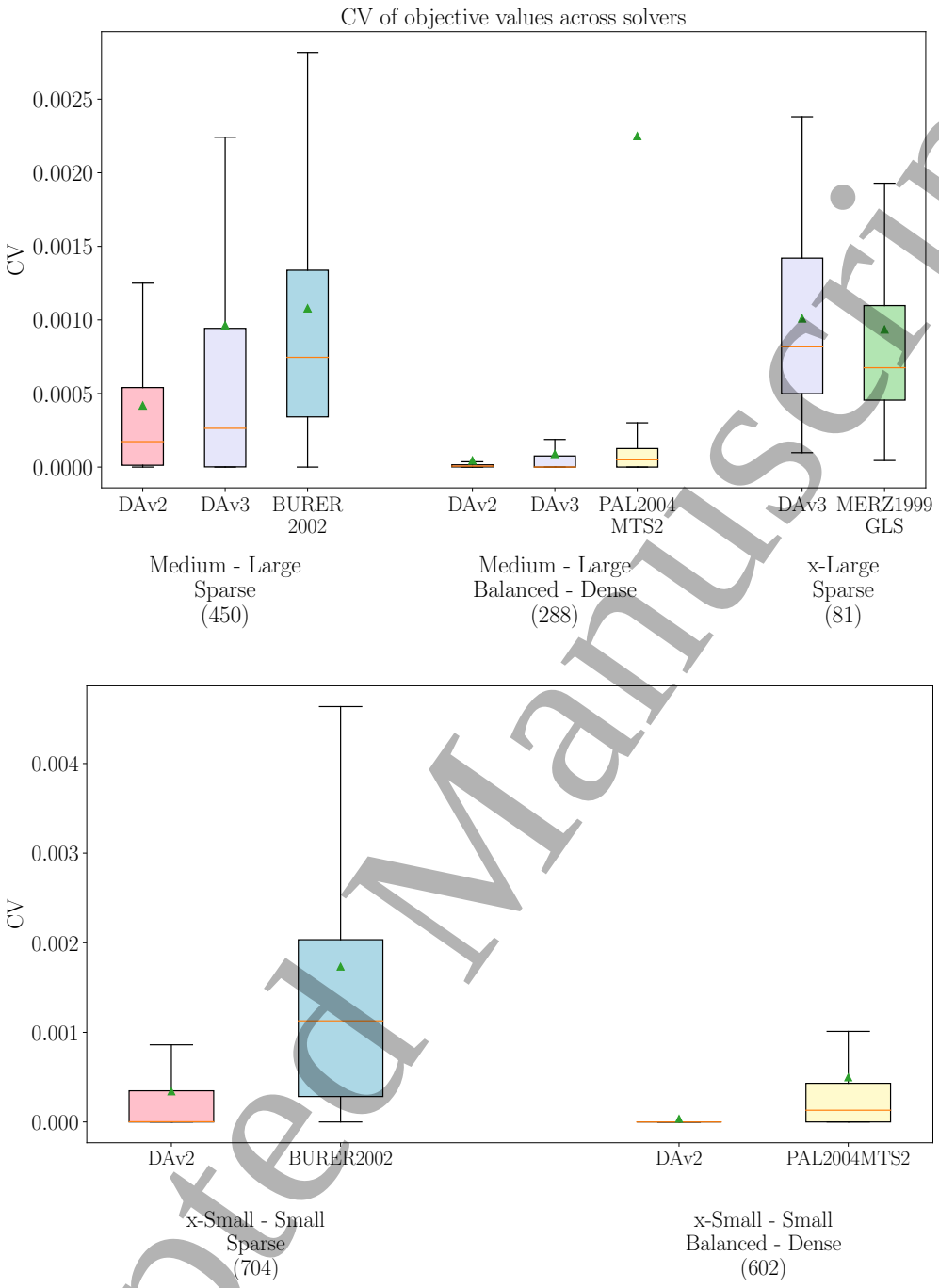
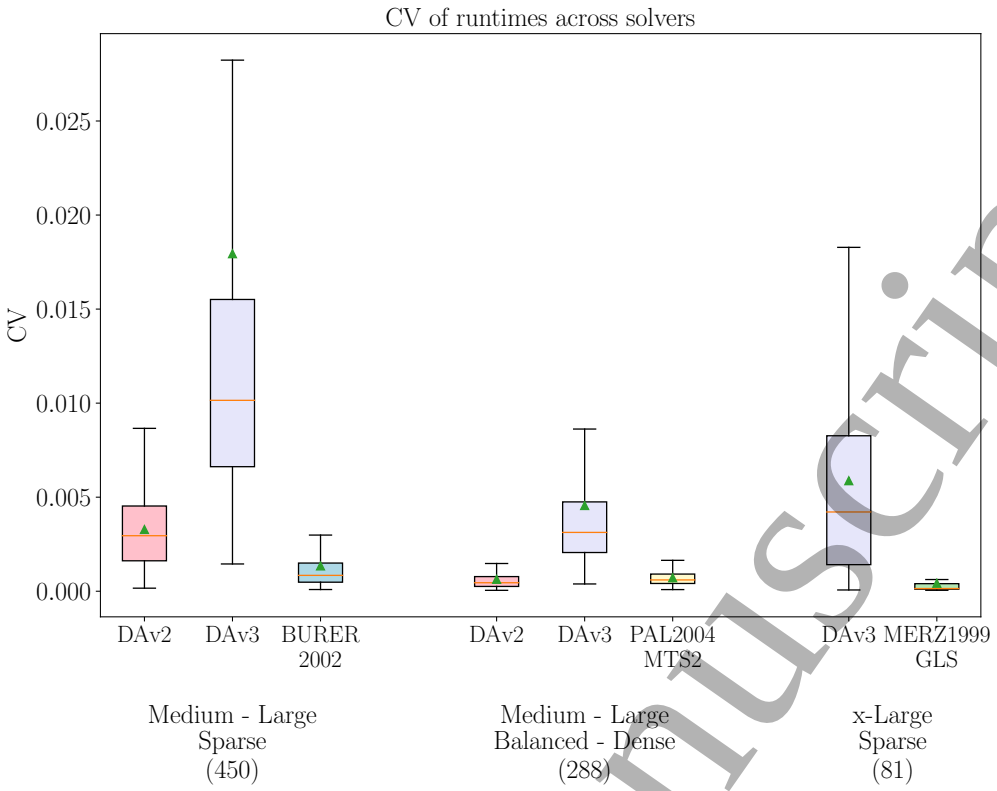


Figure 12: Box plots of the coefficient of variation (CV) of objective values across five independent runs, aggregated over instances in each size-density category studied in Section 4.1 (above) and in Appendix A (below). The number of instances per category indicated in parentheses. Lower CV values indicate lower run-to-run variability of the solver's solution. The mean value is indicated by a green triangle and the median by an orange horizontal line. For clarity of presentation, outliers are not displayed.



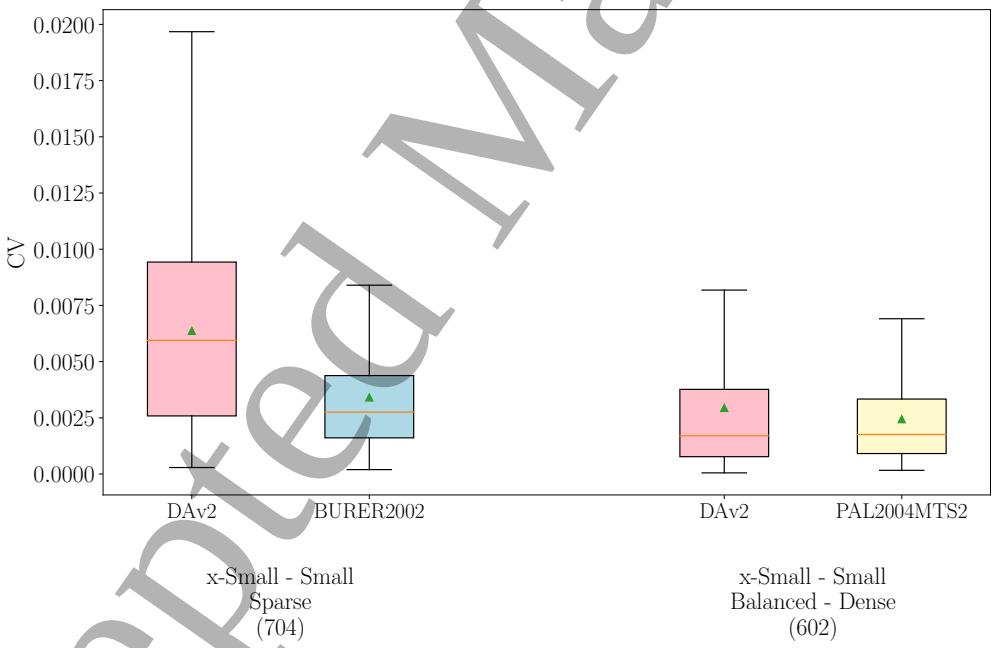


Figure 13: Box plots of the coefficient of variation (CV) of runtimes across five independent runs, aggregated over instances in each size-density category studied in Section 4.1 (above) and in Appendix A (below). The number of instances per category indicated in parentheses. Lower CV values indicate lower run-to-run variability of the solver's runtimes. The mean value is indicated by a green triangle and the median by an orange horizontal line. For clarity of presentation, outliers are not displayed.

D. DAv2 runtime estimation

To allow for DAv2 to terminate within a specific time range, for the purpose explained in Section 3.1, we have obtained fitted functions to estimate the annealing time and CPU time based on the input number of variables, runs, and iterations. To obtain these estimates, we generated runtime data of DAv2 across all instances using various combinations of run and iteration numbers. The number of runs was varied from minimum to maximum in steps of 16, i.e. $16, 32, 48, 64, 80, 96, 112, 128$. The number of iterations was set in relation to the number of variables as $f \times n^2$ where $f \in \{1, 2, 4\}$.

The runtime strongly depends on the size category introduced in Section 3.2 to which a given instance belongs. Accordingly, the annealing time and CPU time functions are defined as

$$\text{Annealing\_time}(\text{runs}, \text{iterations}) = a(\text{runs} \times \text{iterations}) + b \tag{7}$$

$$\text{CPU\_time}(\text{runs}, n) = c(n^2 \times \text{runs}) + d(n \times \text{runs}) + en^2 + k(\text{runs}) + gn + h \tag{8}$$

where the fitting parameters a, b, c, d, e, k, g, h, are given in Table 7 for each size category. We used these equations to estimate the number of runs and iterations required for each instance to as much as possible meet a specified time limit. Specifically, we aimed to select the largest combination of run and iteration numbers that remains within 10% of the time limit.

The procedure is as follows: we begin with the minimum values–16 runs and $\max(10,000, n^2)$ iterations. We first estimate the CPU time under these

	x-Small: $20 \leq n < 1024$	Small: $1024 \leq n < 2048$
a	2.0081 $\times 10^{-6}$	2.0017 $\times 10^{-6}$
b	13.2942	48.6364
c	-0.5576 $\times 10^{-7}$	6.2894 $\times 10^{-7}$
d	0.0007	-0.0007
e	2.9877 $\times 10^{-6}$	3.5768 $\times 10^{-6}$
k	-0.0101	0.7396
g	-0.0020	0.0056
h	4.3422	5.3949
	Medium: $2048 \leq n < 4096$	Large: $4096 \leq n < 8192$
a	2.0010 $\times 10^{-6}$	2.0005 $\times 10^{-6}$
b	193.8656	126.2170
c	7.8667 $\times 10^{-7}$	7.4802 $\times 10^{-7}$
d	-0.0015	-0.0002
e	4.1800 $\times 10^{-6}$	-9.6817 $\times 10^{-6}$
k	1.8767	-3.7253
g	-0.0056	0.1548
h	59.8780	-264.9900

Table 7: Parameters of DAv2 annealing and CPU time fitted functions, Eqs. (7) and (8), respectively.

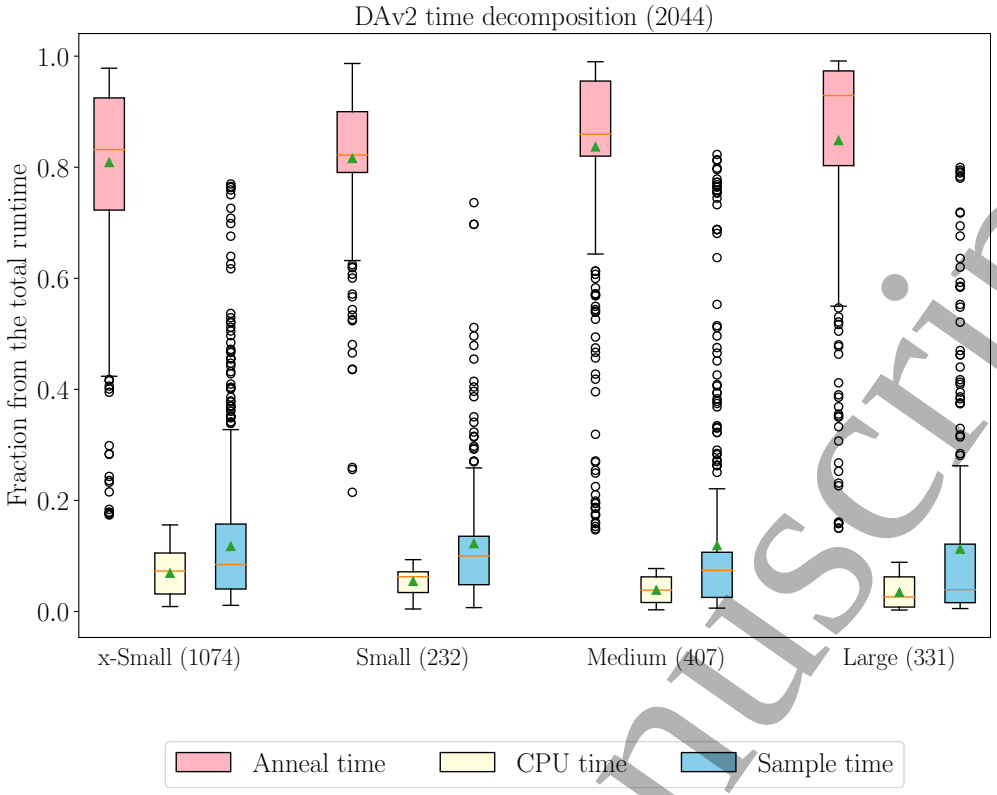


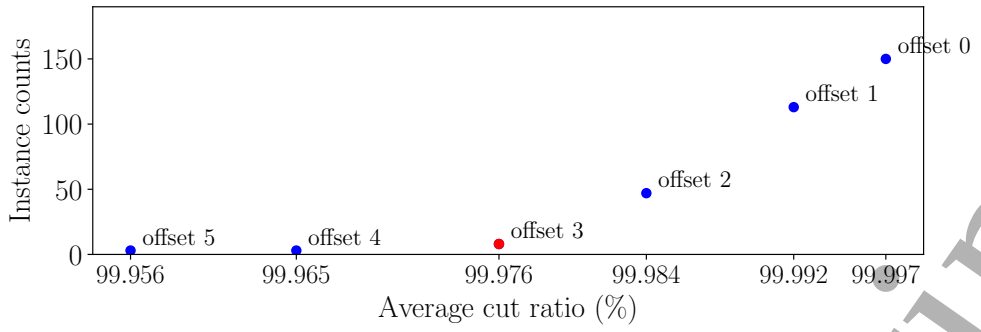
Figure 14: Distribution of relative contributions of DAv2 primary runtime components: anneal, CPU, and sample times, to the total runtime across 2,044 instances. The mean value is indicated by a green triangle and the median by an orange horizontal line.

values using Eq. (8). From 99% of the time limit, we subtract the estimated CPU time along with other fixed overheads, such as sampling[3] time and scaling[4] time to determine the remaining time available for annealing. The 99% threshold allows for a buffer, and the sampling and scaling times are instance-specific constants independent from the number of runs and iterations and are extracted from the collected runtime data. Based on this, we estimate the possible number of iterations using Eq. (7). If this value is more than the previous number of iterations, we update the iteration number accordingly. If it exceeds 40,000 iterations and the estimated total time (CPU time + annealing time + fixed overheads) is still below 99% of the time limit, higher runs are explored. Otherwise, higher runs are not considered further.

To illustrate the distribution of the relative contributions of DAv2's primary runtime components to the total runtime, we present Fig. 14 on 2,044 instances, grouped by instance size categories. The three primary components are anneal time, CPU time, and sample time. Recall that throughout this study, we report the total runtime, which includes these three components

[3] The time for setting the start and end temperature of the annealing process on DAv2. We apply the sampling procedure described in [53] to achieve flip probability of 99% in the beginning and 1% after half of the annealing time.

[4] Rescaling the QUBO matrix by multiplying it with a factor. This step is essential for converting floating-point elements to integers and subsequently adjusting the objective value.



Offset (seconds)	0	1	2	3	4	5
Instance counts with runtime > 10% of limit	150	113	47	8	3	3
Average cut ratio (%)	99.9971	99.9916	99.9838	99.9760	99.9645	99.9560

Figure 15: Trade-off between average cut ratio and time-limit adherence across different offset values for DAv3. An offset of 3 seconds (red point) offers the best balance between runtime compliance and cut ratio quality.

as well as additional overheads discussed earlier. As shown in the figure, anneal time typically constitutes the largest fraction of the total runtime. Some outliers in the low anneal-time fraction arise because the annealing time is allocated as the remaining time after accounting for the other time components. This pattern is consistent across all instance size categories and remains stable for each of the three runtime components.

E. DAv3 time limit offset determination

As noted in Section 3.1, DAv3 is more prone to far exceed shorter time limits, so our focus is on identifying an appropriate time offset specifically for such cases. Due to its stochastic runtime behavior, we were unable to obtain a fitted runtime function as done for DAv2 in Appendix D. Instead, we determine a suitable time limit offset empirically. To this end, we executed DAv3 on all Medium to x-Large instances that have been assigned by the algorithm in Section 3.1 time limits between 0.25 and 100 seconds, with an offset-adjusted time limit defined in Eq. (5) with various offsets: 0 to 5 seconds in 1-second increments. Our goal is to identify the offset that minimizes the number of instances exceeding the 10% time safety margin while achieving the highest possible average cut ratio. The average cut ratio for each offset is computed as

$$\text{Average\_cut\_ratio}(\%) =: \frac{1}{N}\left(\sum_{i=1}^{N}\frac{\text{cut}_i}{\text{best}_i}\right)\times 100 \tag{9}$$

where cut_i is the achieved objective value for instance i for each corresponding offset, best_i is the best objective value found across all available offset runs for instance i, and $N = 819$ is the total number of instances in the Medium to x-Large categories. In Fig. 15, we show how the offset affects the average cut ratio and the number of instances whose runtime exceeds 10% of the time

limit. An offset of 3 seconds fairly reduces the number of instances that violate the safety margin, with only a marginal decrease in average cut ratio. Higher offsets yield minimal improvements but come with additional quality losses, making them less favorable.

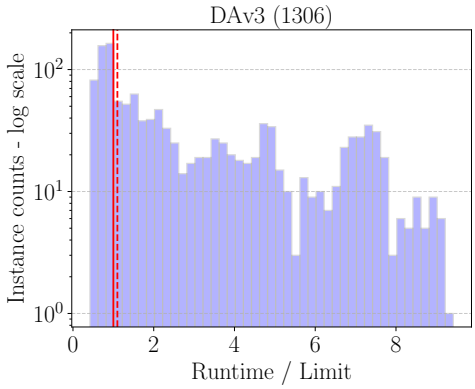
Unfortunately, an explicit decomposition of DAv3's runtime into different time components is not accessible; therefore, a plot analogous to Fig. 14 for DAv2 is not possible.

F. Runtime deviation of solvers from baseline time limit

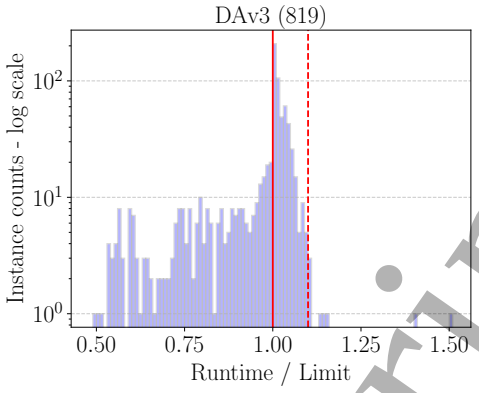
Solvers do not typically terminate exactly at the specified time limit. To account for this, we define a safety margin with an upper bound of 10%. In Fig. 16, we present histograms showing instance counts by each solver's runtime-to-limit ratio. As shown in Fig. 16c, the runtime fitting of DAv2, detailed in Appendix D, was largely successful. However, it also occasionally resulted in a runtime shorter than the assigned limit. It still exceeds the time limit slightly in some cases, but it is still well below the 10% upper limit of the safety margin.

Although DAv3 includes a time limit termination feature and an added buffer, it still exceeds the safety margin on a significant number of instances–especially those in the x-Small and Small categories (see Fig. 16a). This is partially due to a 1-second minimum runtime, as well as the communication overhead between the DAU and the software layer, making it more likely to exceed the specified time limit. As a result, we restrict our analysis for DAv3 to larger instances, the raw data are nevertheless provided in the accompanying GitHub repository [23]. The time analysis for DAv3 is presented in Fig. 16b to the Medium through x-Large categories. For a large number of instances, the runtime of DAv3 is far below the time limit–more so than DAv2–with runtimes dropping to as low as 50% of the limit. This could partly explain why DAv2 provides a higher 'win' ratio than DAv3.

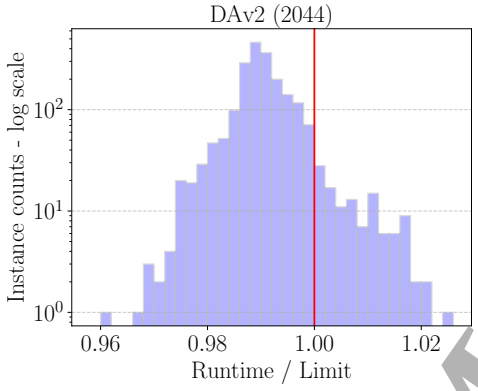
As expected, MQLib solvers also often exceed the time limit as shown in Figs. 16d to 16f. In particular, BURER2002 and PAL2004bMTS2 exceeded the safety margin for multiple instances. Overall, the DA runtimes remain on average lower than those of the MQLib heuristics.



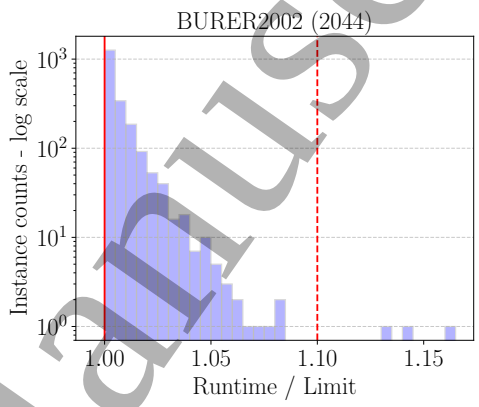
(a) x-Small to Small categories (not analyzed).



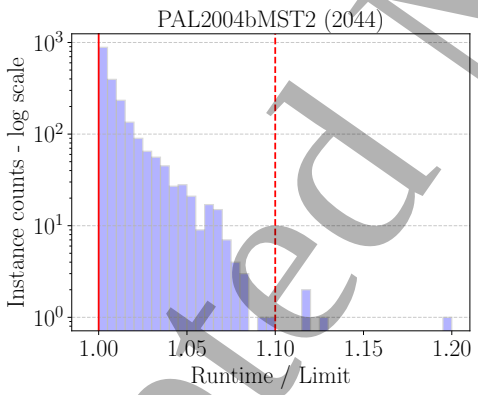
(b) Medium to x-Large categories.



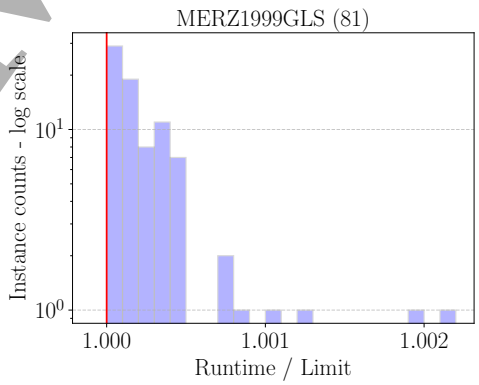
(c) x-Small to Large categories.



(d) x-Small to Large categories.



(e) x-Small to Large categories.



(f) x-Large category.

Figure 16: Histogram (log scale) of relevant instance counts distributed over solver runtime-to-limit ratio. The solid red line at 1 indicates the point where runtime equals the time limit, while the dashed red line marks a 10% exceedance margin that we enforced for our solvers on the majority of instances. DAv2 and MERZ1990GLS remain within this margin, whereas DAv3 in (b), BURER2002, and PALUBECKIS2004bMTS2 exceed it for very few instances. Panel (a) illustrates why the corresponding DAv3 results were excluded from the analysis.

Acronyms

CV coefficient of variation . . . 34
DAU Digital Annealer Unit . . . 8
HS hybrid solver . . . 6
Max-Cut maximum cut . . . 3
MQLib Max-Cut and QUBO instances library . . . 6
QAOA quantum approximate optimization algorithm . . . 4
QUBO quadratic unconstrained binary optimization . . . 2
DA Digital Annealer . . . 2

References

[1] F. Glover, G. A. Kochenberger, and Y. Du, *Applications and computational advances for solving the QUBO model*, in *Springer eBooks* (2022) p. 39–56.

[2] A. Lucas, *Ising formulations of many NP problems*, Frontiers in Physics **2**, 5 (2014), arXiv:1302.5843 [cond-mat.stat-mech].

[3] F. Glover, G. Kochenberger, and Y. Du, *A Tutorial on Formulating and Using QUBO Models*, arXiv:1811.11538 [cs.DS] (2018).

[4] D. Ratke, *List of QUBO formulations*, `https://blog.xa0.de/post/List-of-QUBO-formulations/` (2021), accessed 2025-06-19.

[5] A. P. Punnen, ed., *The Quadratic Unconstrained Binary Optimization Problem: Theory, Algorithms, and Applications*, 1st ed. (Springer Cham, 2022) pp. XIII + 319, eBook published July 12, 2022.

[6] S. Aaronson, *The limits of quantum computers*, Scientific American (2008).

[7] H. Goto, K. Tatsumura, and A. R. Dixon, *Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems*, Science Advances **5**, eaav2372 (2019), https://www.science.org/doi/pdf/10.1126/sciadv.aav2372.

[8] K. Tatsumura, M. Yamasaki, and H. Goto, *Scaling out Ising machines using a multi-chip architecture for simulated bifurcation*, Nature Electronics **4**, 208 (2021).

[9] D-Wave Quantum Inc., *Performance gains in the D-Wave Advantage2 system at the 4,400-qubit scale*, Whitepaper 14-1083A-A (D-Wave Quantum Inc., 2025) released May 12, 2025.

[10] T. Okuyama, T. Sonobe, K.-i. Kawarabayashi, and M. Yamaoka, *Binary optimization by momentum annealing*, Phys. Rev. E **100**, 012111 (2019).

[11] T. Inagaki, Y. Haribara, K. Igarashi, T. Sonobe, S. Tamate, T. Honjo, A. Marandi, P. L. McMahon, T. Umeki, K. Enbutsu, O. Tadanaga, H. Takenouchi, K. Aihara, K. ichi Kawarabayashi, K. Inoue, S. Utsunomiya, and H. Takesue, *A coherent Ising machine for 2000-node optimization problems*, Science **354**, 603 (2016).

[12] T. Honjo, T. Sonobe, K. Inaba, T. Inagaki, T. Ikuta, Y. Yamada, T. Kazama, K. Enbutsu, T. Umeki, R. Kasahara, K. ichi Kawarabayashi, and H. Takesue, *100,000-spin coherent Ising machine*, Science Advances **7**, eabh0952 (2021).

[13] H. Nakayama, J. Koyama, N. Yoneoka, and T. Miyazawa, *Third Generation Digital Annealer Technology*, Tech. Rep. (Fujitsu Laboratories, 2021) accessed: 2025-07-23.

[14] N. Mohseni, P. L. McMahon, and T. Byrnes, *Ising machines as hardware solvers of combinatorial optimization problems*, Nature Reviews Physics **4**, 363 (2022), arXiv:2204.00276 [quant-ph].

[15] M. X. Goemans and D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, Journal of the ACM (JACM) **42**, 1115 (1995).

[16] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell, *Optimal inapproximability results for MAX-CUT and other 2-variable CSPs?*, SIAM Journal on Computing **37**, 319 (2007).

[17] I. Dunning, S. Gupta, and J. Silberholz, *What works best when? a systematic evaluation of heuristics for Max-Cut and QUBO*, INFORMS J. on Computing **30**, 608–624 (2018).

[18] F. Phillipson, *Fair benchmarking combinatorial optimization solvers in the era of emerging computing paradigms*, in *Innovations for Community Services*, Communications in Computer and Information Science, Vol. 2513, edited by S. Zielinski, G. Eichler, C. Erfurth, and G. Fahrnberger (Springer, United States, 2025) pp. 79–93, data source:; 25th International Conference on Innovations for Community Services, I4CS 2025, I4CS 2025; Conference date: 11-06-2025 Through 13-06-2025.

[19] D-Wave Systems Inc., *Hybrid solver service: An overview*, https://www.dwavequantum.com/media/4bnpi53x/14-1039a-b_d-wave_hybrid_solver_service_an_overview.pdf (2020), white Paper.

[20] J. Yang, D. Wang, X. Zhao, H. Zhang, M. Gao, and L. Yang, *A Novel Solver for QUBO Problems: Performance Analysis and Comparative Study with State-of-the-Art Algorithms*, arXiv:2506.04596 [quant-ph] (2025).

[21] I. Dunning, S. Gupta, and J. Silberholz, *Mqlib: Implementations of*

heuristics for Max-Cut and QUBO (2018), accessed: 2023-12-11. Git commit `686a4fc`.

[22] Y. Ye, *Gset collection of graphs for Max-Cut benchmarking*, https://web.stanford.edu/~yyye/yyye/Gset/ (2003), accessed 2025-06-26.

[23] S. Shaglel and M. Kirsch, *DA Maxcut Benchmark (DAMB)*, https://github.com/SalwaShaglel/DAMB (2025).

[24] E. Farhi, J. Goldstone, and S. Gutmann, *A Quantum Approximate Optimization Algorithm*, arXiv:1411.4028 [quant-ph] (2014).

[25] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, *Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices*, Phys. Rev. X **10**, 021067 (2020).

[26] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, *A review on quantum approximate optimization algorithm and its variants*, Physics Reports **1068**, 1 (2024), a review on Quantum Approximate Optimization Algorithm and its variants.

[27] E. Farhi, J. Goldstone, and S. Gutmann, *A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem*, arXiv:1412.6062 [quant-ph] (2014).

[28] B. Barak, A. Moitra, R. O'Donnell, P. Raghavendra, O. Regev, D. Steurer, L. Trevisan, A. Vijayaraghavan, D. Witmer, and J. Wright, *Beating the random assignment on constraint satisfaction problems of bounded degree*, arXiv:1505.03424 [cs.CC] (2015).

[29] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, *Barren plateaus in quantum neural network training landscapes*, Nature Communications **9**, 10.1038/s41467-018-07090-4 (2018).

[30] L. Bittel and M. Kliesch, *Training Variational Quantum Algorithms Is NP-Hard*, Phys. Rev. Lett. **127**, 120502 (2021), arXiv:2101.07267 [quant-ph].

[31] L. Bittel, S. Gharibian, and M. Kliesch, *Optimizing the depth of variational quantum algorithms is strongly QCMA-hard to approximate*, in *38th Computational Complexity Conference (CCC 2023)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 264, edited by A. Ta-Shma (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2023) pp. 34:1–34:24, arXiv:2211.12519 [quant-ph] .

[32] G. G. Guerreschi and A. Y. Matsuura, *QAOA for Max-Cut requires hundreds of qubits for quantum speed-up*, Scientific Reports **9**, 10.1038/s41598-019-43176-9 (2019).

[33] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J.

Coles, *Noise-induced barren plateaus in variational quantum algorithms*, Nature Communications **12**, 10.1038/s41467-021-27045-6 (2021).

[34] Y. R. Sanders, D. W. Berry, P. C. Costa, L. W. Tessler, N. Wiebe, C. Gidney, H. Neven, and R. Babbush, *Compilation of fault-tolerant quantum heuristics for combinatorial optimization*, PRX Quantum **1**, 020312 (2020).

[35] Z. Cai, R. Babbush, S. C. Benjamin, S. Endo, W. J. Huggins, Y. Li, J. R. McClean, and T. E. O'Brien, *Quantum error mitigation*, Reviews of Modern Physics **95**, 045005 (2023), arXiv:2210.00921 [quant-ph].

[36] R. Hamerly, T. Inagaki, P. L. McMahon, D. Venturelli, A. Marandi, T. Onodera, E. Ng, C. Langrock, K. Inaba, T. Honjo, K. Enbutsu, T. Umeki, R. Kasahara, S. Utsunomiya, S. Kako, K.-i. Kawarabayashi, R. L. Byer, M. M. Fejer, H. Mabuchi, D. Englund, E. Rieffel, H. Takesue, and Y. Yamamoto, *Experimental investigation of performance differences between coherent ising machines and a quantum annealer*, Science Advances **5**, 10.1126/sciadv.aau0823 (2019).

[37] K. Boothby, P. Bunyk, J. Raymond, and A. Roy, *Next-generation topology of d-wave quantum processors* (2020), arXiv:2003.00133 [quant-ph] .

[38] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, *Perspectives of quantum annealing: methods and implementations*, Reports on Progress in Physics **83**, 054401 (2020).

[39] C. D. Gonzalez Calaza, D. Willsch, and K. Michielsen, *Garden optimization problems for benchmarking quantum annealers*, Quantum Information Processing **20**, 10.1007/s11128-021-03226-6 (2021).

[40] D. Willsch, M. Willsch, C. D. Gonzalez Calaza, F. Jin, H. De Raedt, M. Svensson, and K. Michielsen, *Benchmarking advantage and d-wave 2000q quantum annealers with exact cover problems*, Quantum Information Processing **21**, 10.1007/s11128-022-03476-y (2022).

[41] S. Matsubara, M. Takatsu, Toshiyuki Miyazawa, T. Miyazawa, T. Shibasaki, Y. Watanabe, K. Takemoto, and H. Tamura, *Digital Annealer for High-Speed Solving of Combinatorial optimization Problems and Its Applications*, Asia and South Pacific Design Automation Conference , 667.

[42] T. Ikuta *et al.*, *Solving the maximum cut benchmark with an optimization solver*, Research Institute for Mathematical Sciences Kokyuroku **1941**, 49 (2015), (In Japanese).

[43] F. Ma and J.-K. Hao, *A multiple search operator heuristic for the max-k-cut problem* (2015), arXiv:1510.09156 [cs.DM] .

[44] T. Huang, J. Xu, T. Luo, X. Gu, R. Goh, and W.-F. Wong, *Benchmark-*

ing quantum(-inspired) annealing hardware on practical use cases, IEEE Transactions on Computers **72**, 1692 (2023).

[45] H. Oshiyama and M. Ohzeki, *Benchmark of quantum-inspired heuristic solvers for quadratic unconstrained binary optimization*, Scientific Reports **12**, 10.1038/s41598-022-06070-5 (2022).

[46] O. Şeker, N. Tanoumand, and M. Bodur, *Digital annealer for quadratic unconstrained binary optimization: A comparative performance analysis*, Appl. Soft Comput. **127**, 10.1016/j.asoc.2022.109367 (2022).

[47] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual* (2023).

[48] T. Achterberg, *SCIP: solving constraint integer programs*, Mathematical Programming Computation **1**, 1 (2009).

[49] S. Burer, R. D. C. Monteiro, and Y. Zhang, *Rank-two relaxation heuristics for max-cut and other binary quadratic programs*, SIAM J. Optim. **12**, 503 (2002).

[50] G. Palubeckis, *Multistart tabu search strategies for the unconstrained binary quadratic optimization problem*, Annals of Operations Research **131**, 259 (2004).

[51] J.-R. Jiang, Y.-C. Shu, and Q.-Y. Lin, *Benchmarks and recommendations for quantum, digital, and GPU annealers in combinatorial optimization*, IEEE Access **12**, 125014 (2024).

[52] M. Kowalsky, T. Albash, I. Hen, and D. A. Lidar, *3-regular three-XORSAT planted solutions benchmark of classical and quantum heuristic optimizers*, Quantum Science and Technology **7**, 025008 (2022), arXiv:2103.08464 [quant-ph].

[53] C. Münch, F. Schinkel, S. Zielinski, and S. Walter, *Transformation-Dependent Performance-Enhancement of Digital Annealer for 3-SAT*, arXiv e-prints , arXiv:2312.11645 (2023), arXiv:2312.11645 [quant-ph].

[54] Y.-T. Kao, J.-L. Liao, and H.-C. Hsu, *Solving Combinatorial Optimization Problems on Fujitsu Digital Annealer*, arXiv:2311.05196 [quant-ph] (2023).

[55] D. Leib, T. Seidel, S. Jäger, R. Heese, C. Jones, A. Awasthi, A. Niederle, M. Bortz, and et al., *An optimization case study for solving a transport robot scheduling problem on quantum-hybrid and quantum-inspired hardware*, Scientific Reports **13**, 18743 (2023).

[56] C. Lee, P.-H. Wang, and Y. J. Tseng, *Digital annealing optimization for natural product structure elucidation*, Briefings in Bioinformatics **25**, bbae600 (2024).

[57] A. A. Jha, E. L. Stoyanoff, G. Khundzakishvili, P. Kairys, H. Ushijima-Mwesigwa, and A. Banerjee, *Digital annealing route to complex magnetic phase discovery*, in *2021 International Conference on Rebooting Computing (ICRC)* (2021) pp. 119–123.

[58] A. Maruo, H. Igarashi, H. Oshima, and S. Shimokawa, *Optimization of planar magnet array using digital annealer*, IEEE Transactions on Magnetics **56**, 1 (2020).

[59] J. Dornemann, S. Shaglel, M. Kliesch, and A. Taraz, *A hybrid quantum-inspired and deep learning approach for the capacitated vehicle routing problem with time windows*, in *THE 19TH LEARNING AND INTELLIGENT OPTIMIZATION CONFERENCE* (2025).

[60] H. Cohn and M. Fielding, *Simulated annealing: Searching for an optimal temperature schedule*, SIAM Journal on Optimization **9**, 779 (1999), https://doi.org/10.1137/S1052623497329683.

[61] S. Chen, J. S. Rosenthal, A. Dote, H. Tamura, and A. Sheikholeslami, *Optimization via rejection-free partial neighbor search*, Statistics and Computing **33**, 131 (2023).

[62] S. Chen, J. S. Rosenthal, A. Dote, H. Tamura, and A. Sheikholeslami, *Sampling via rejection-free partial neighbor search*, Communications in Statistics - Simulation and Computation **54**, 837 (2025), https://doi.org/10.1080/03610918.2023.2266157.

[63] A. Lipowski and D. Lipowska, *Roulette-wheel selection via stochastic acceptance*, Physica A: Statistical Mechanics and its Applications **391**, 2193 (2012).

[64] Fujitsu, *Digital annealer api documentation*, `https://portal.aispf.global.fujitsu.com/apidoc/da/en/index.html`, accessed: 2025-07-22.

[65] J. Beasley, *Or-library: A collection of test instances for or problems* (1990).

[66] T. Koch, A. Martin, D. Rehfeldt, and S. Voss, *Steinlib: A library for steiner tree problems*.

[67] D. Johnson and M. Trick, eds., *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science (1993).

[68] *Seventh DIMACS implementation challenge: Semidefinite and related optimization problems* (2000).

[69] *11th dimacs implementation challenge: Steiner tree problems* (2013).

[70] G. Reinelt, *TSPLIB–a traveling salesman problem library*, ORSA Journal on Computing **3**, 376–384 (1991).

[71] A. Wiegele, *Biq Mac library: Max-Cut and binary quadratic programming instances* (2007), documentation available at https://biqmac.aau.at/biqmaclib.pdf.

[72] J. Culberson, *Flat graph generator*.

[73] A. Hagberg, P. Swart, and D. Schult, *NetworkX – Python package for complex network analysis* (2008).

[74] G. Rinaldi, *Rudy: A Rudimental Graph Generator* (1995).

[75] P. Merz and B. Freisleben, *Genetic algorithms for binary quadratic programming*, in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, GECCO'99 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999) p. 417–424.

[76] J. Vodeb, V. Eržen, T. Hrga, and J. Povh, *Accuracy and Performance Evaluation of Quantum, Classical and Hybrid Solvers for the Max-Cut Problem*, arXiv:2412.07460 [math.OC] (2024).