Boosting Quantum Annealing Performance using Evolution Strategies for Annealing Offsets Tuning

Sheir Yarkoni[1,2], Hao Wang[2], Aske Plaat[2], and Thomas Bäck[2]

[1] D-Wave Systems Inc., Burnaby, Canada
[2] LIACS, Leiden University, Netherlands

Abstract. In this paper we introduce a novel algorithm to iteratively tune annealing offsets for qubits in a D-Wave 2000Q quantum processing unit (QPU). Using a (1+1)-CMA-ES algorithm, we are able to improve the performance of the QPU by up to a factor of 12.4 in probability of obtaining ground states for small problems, and obtain previously inaccessible (i.e., better) solutions for larger problems. We also make efficient use of QPU samples as a resource, using 100 times less resources than existing tuning methods. The success of this approach demonstrates how quantum computing can benefit from classical algorithms, and opens the door to new hybrid methods of computing.

Keywords: Quantum computing · Quantum annealing · Optimization · Hybrid algorithms

1 Introduction

Commercial quantum processing units (QPUs) such as those produced by D-Wave Systems have been the subject of many characterization tests in a variety of optimization, sampling, and quantum simulation applications [1–6]. However, despite the increasing body of work that showcases the various uses of such QPUs, application-relevant software that uses the QPUs intelligently has been slow to develop. This can be most easily attributed to the fact that these QPUs have a multitude of parameters, with each contributing (possibly in a co-dependent manner) to the performance of the processor. In practice it is intractable to tune all parameters, and often a subset of parameters are chosen to be explored in detail. In this paper we introduce the use of a Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to heuristically tune one set of parameters, the so-called annealing offsets, on-the-fly, in an application-agnostic manner. We test the viability and performance of our tuning method using randomly generated instances of the Maximum Independent Set (MIS) problem, a known NP-hard optimization problem that has been tested on a D-Wave QPU before [7]. The definition of the MIS problem is as follows: given a graph G composed of vertices V and edges E between them, find the largest subset of nodes $V' \subseteq V$ such that no two nodes in V' contain an edge in E. The tuning method introduced in this paper does not exploit the nature of the MIS problem nor makes any assumptions regarding the structure of the associated graph, and can be used for problem sets other than MIS instances.

The QPUs available from D-Wave Systems implement a quantum annealing algorithm which samples from a user-specified Ising Hamiltonian (in a $\{-1, +1\}$ basis) or equivalently a quadratic unconstrained binary optimization (QUBO) problem (in a $\{0, 1\}$ basis) [8]. Minimizing a QUBO or Ising Hamiltonian is known to be an NP-hard problem [9], so many interesting and difficult computational problems can be formulated as Ising models and QUBOs [10]. A QUBO problem and its associated objective function can be represented as the following:

$$\mathrm{Obj}(x) = x^T \cdot Q \cdot x, \tag{1}$$

where Q is a matrix $\in \mathbb{R}^{N\times N}$, and x is a binary string $\in \{0,1\}^N$.

The rest of the paper is organized as follows. In the next section we introduce the previous works regarding annealing offsets in D-Wave QPUs. Section 3 describes the motivation and use of annealing offsets as implemented by D-Wave. In Section 5 we explain in detail the algorithm implemented in this paper: how the annealing offsets were evolved and then used to solve randomly generated MIS problems. The results obtained using this algorithm are shown in Section 6, and are followed by conclusions in Section 7.

2 Previous works

It has been shown experimentally in [11] that the probability of staying in the ground state of the quantum system can be improved by applying annealing offsets to small qubit systems. Specifically, the authors use a first-order perturbative expansion of the quantum annealing Hamiltonian to connect ground state degeneracy and single-spin floppiness to change the annealing offset for particular qubits. Small instances were chosen (24 qubits) such that exact diagonalization of the quantum Hamiltonian was possible, which is necessary for the iterative algorithm presented in [11]. For these instances, which were chosen specifically to be difficult for quantum annealing, median success probability of the QPU was improved from 62% to 85% using an iterative algorithm. In a more recent white paper produced by D-Wave Systems [12], it was shown that for specific inputs sets it is possible to boost performance of the QPU by up to a factor of 1000. These results were obtained by using a grid-search technique, requiring 2.5 million samples to be generated using the QPU for a single input problem. The model used to adjust the annealing offsets in this paper was similar to the model used in [11]. This technique was also applied to 2-SAT problems with 12 Boolean variables in [13].

It has been shown (using quantum Monte Carlo simulations) that annealing offsets can mitigate first-order phase transitions, exponentially enhancing performance when compared to simulated thermal annealing [14]. However, this work used simple problems that could be solved exactly. Understanding how this asymptotic behavior affects NP-hard problems in general is an open question. Previous work regarding solving maximum independent set problems using a D-Wave 2000Q QPU shows that the QPU performance suffers greatly as problem sizes increase, even as problems become easier to solve classically [7]. The new annealing offset features in the D-Wave 2000Q processor are designed to mitigate the likely sources of this performance degradation seen in [7]. Tuning these parameters using a CMA-ES routine is the subject of this work.

3 Quantum annealing offset parameters

In uniform quantum annealing as implemented in D-Wave QPUs [8,15], all qubits begin their evolution at the same point in time. Consider an annealing procedure defined as follows:

$$H(s) = A(s)\sum_i \sigma_i^x + B(s)\left[\sum_i h_i\sigma_i^z + \sum_{ij} J_{ij}\ \sigma_i^z \otimes \sigma_j^z\right],$$

where A(s) is the initial Hamiltonian driver function (transverse field), B(s) is the target Hamiltonian driver function (terminal energy scale), h and J define the target Hamiltonian, s is a normalized time parameter (real time t divided by total anneal time τ), and σ_i^x and σ_i^z are the x and z Pauli spin matrices operating on the ith qubit respectively. By default, all qubits will begin their annealing procedure at time $s = 0$, and terminate at $s = 1$. However, a feature in the D-Wave 2000Q processor allows users to set an advance or delay for qubits in physical time, and in an independent manner. Meaning, each qubit can have its evolution path advanced/delayed by some user specified[3] Δs. Thus the functions A(s) and B(s) are now an ensemble of functions, A(s $+\Delta s_i$) and B(s $+\Delta s_i$), ranging from $s = 0 + \Delta s_i$ to $s = 1 + \Delta s_i$, and can differ from qubit to qubit.

The benefit of adding this additional control feature is motivated by the physics governing quantum annealing. As demonstrated in [4], the distribution of answers produced by the QPU are more than a result of the finite temperature Boltzmann distribution, but are also an artifact of ergodicity breaking during the annealing procedure, and are a result of the annealing path as defined by the A(s) and B(s) functions in Equation 3. Allowing a programmable method to delay and advance the anneal schedule of individual qubits allows some mitigation of this effect, sometimes called "freeze-out". It has even been shown that, for careful construction of these offsets on a per-qubit basis, it may be possible to avoid forbidden crossings entirely [14]. However, this requires *a priori* knowledge of the energy landscape, and is thus computationally impractical for most problems.

[3] Each qubit in the QPU has a different range in Δs that can be set independently. $\Delta s < 0$ is an advance in time and $\Delta s > 0$ is a delay. A typical range for Δs is $\pm$ 0.15. The total annealing time for all qubits is still $|s| = 1$ (or τ in units of time).

4 Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [16] is a state-of-the-art stochastic optimization algorithm for the continuous black-box optimization problem. To tune the annealing offsets, we adopt the so-called $(1+1)$-CMA-ES variant [17], which generates only one candidate search point in each iteration. The $(1+1)$-CMA-ES algorithm exhibits both fast convergence and global search ability. The choice of the optimization algorithm is made based on the following considerations: firstly, QPU time is considered an expensive resource, and we wish to spend as little QPU time for the tuning as possible. Secondly, the QPU is a serial device, meaning that problems (a set of annealing offset parameters in our case) can only be tested sequentially. Therefore, the search strategies that generate multiple candidate points are not preferred as it is not possible to parallelize those points in our case.

For the experiments reported in this paper, we run the $(1+1)$-CMA-ES with its default parameter settings, since it is well known that such a setting shows quite robust behaviors across many benchmark functions [18]. Only the method for generating the initial candidate solution is varied in the experiment: we investigate the effects of initializing the annealing offsets either uniformly within the corresponding upper/lower bounds, or with a value of zero.

5 Tuning QA parameters with (1+1)-CMA-ES

In this paper, it is proposed to tune the annealing offsets of a D-Wave QPU with the $(1+1)$-CMA-ES algorithm [17], aiming at improving the performance of the QPU on specific problem classes, for instance the MIS problem. MIS problem instances are constructed using the same method as in [7], using random graphs with edge probability $p = 0.2$ (the empirically determined difficult point for the QPU). The minor-embedding techniques applied to solve these problems directly on the QPU topology are also as in [7]. To solve these problems, the annealing offsets of qubits should be set up properly and thus are tuned using $(1+1)$-CMA-ES. We considered the qubits within each chain to be a single logical qubit, as with the purpose of embedding. Therefore, in calculating the offsets, we consider the *collective* minimum and maximum offsets that can be used on the chain as a whole. Explicitly, we find the highest minimum and lowest maximum shared between all qubits in every chain, and use those as the boundary for each logical qubit (chain). Every qubit within every chain is therefore advanced/delayed by the same amount.

In the tuning experiment, two initialization methods of annealing offsets are compared: *uniform* and *none*. The former is a relatively standard approach: given a feasible range for each offset, $[l_i, u_i]$ for qubits in chain c_i, a random number is sampled uniformly within this range, which is then used as the initial search point for all qubits in the chain. This method allows for a fair exploration of the entire search space. It is especially important in such a quadratic model, where the influences of each annealing offset on the others are not obvious and unpredictable in the worst case. The second method, where all offsets set are set to zero initially (representing annealing offsets), is also tested because it represents a "not bad" initial position, and is also the starting point for the algorithm presented in [11]. This is the default setting when using D-Wave QPUs. We consider this point to be a local optimum for annealing offsets and attempt to improve upon this optimum using the CMA-ES procedure.

The fitness function (objective function) of $(1+1)$-CMA-ES is calculated as the mean energy of the solutions returned by the QPU. This was observed to be the most stable fitness function (as opposed to the 25% percentile, or the minimum energy). Due to the stochastic nature of the samples returned by the QPU, it is important to use a stable metric to evaluate the fitness function in every iteration. According to some preliminary tests, other metrics are too noisy to enable the fast convergence of the $(1+1)$-CMA-ES algorithm. Examples of tuning runs are presented in Appendix A.

Algorithm 1 Tune annealing offsets using $(1+1)$-CMA-ES

```
1: procedure TUNE-OFFSET($\mathbf{l}, \mathbf{u}, B$, StepCost)
2:     Initialize: $\sigma \leftarrow \max\{\mathbf{u}-\mathbf{l}\}/4,\ \mathbf{C}=\mathbf{I},\ c \leftarrow 0$
3:     if InitialOffsets $= 0$ then
4:         $\mathbf{x} \leftarrow \mathbf{0}$                              ▷ offset: zero initialization
5:     else
6:         $\mathbf{x} \leftarrow \mathcal{U}(l_i, u_i)$                 ▷ offset: uniform initialization
7:     $f(\mathbf{x}) \leftarrow$ CALL-QPU($\mathbf{x}$, StepCost)
8:     $\mathbf{A}\mathbf{A}^\top \leftarrow \mathbf{C}$                 ▷ Cholesky decomposition
9:     while $c < B$ do
10:        $\mathbf{z} \leftarrow \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$   ▷ standard normal distribution
11:        $\mathbf{x}' \leftarrow \mathbf{x} + \sigma\mathbf{A}\mathbf{z}$
12:        $f(\mathbf{x}') \leftarrow$ CALL-QPU($\mathbf{x}'$, StepCost)
13:        $\sigma \leftarrow$ UPDATE-STEP-SIZE($\sigma$)
14:        if $f(\mathbf{x}') < f(\mathbf{x})$ then
15:            $\mathbf{x} \leftarrow \mathbf{x}'$
16:            $\mathbf{A} \leftarrow$ UPDATE-CHOLESKY($\mathbf{A}, \mathbf{z}$)
17:        $c \leftarrow c +$ StepCost
18:    return $\mathbf{x}$
```

Pseudocode outlining the algorithm used to tune the offsets is shown in Alg. 1 and the appropriate terms (along with the values used in our experiments, when applicable) are defined in Tab. 1. Essentially, the proposed algorithm optimizes the annealing offsets using the so-called mutation operation (Line 10 and 11), where the current annealing offset $\mathbf{x}$ is perturbed by a Gaussian random vector $\sigma\mathbf{A}\mathbf{z}$ (which is transformed and rescaled from the standard normal vector $\mathbf{z}$, Line 10). The resulting mutation $\mathbf{x}'$ is evaluated in the QPU (Line 12) and it is passed onto the next iteration if its objective value $f(\mathbf{x}')$ is better than $f(\mathbf{x})$ (Line 14 and 15). In addition, two procedures, UPDATE-STEP-SIZE and UPDATE-CHOLESKY are adopted to control the step-size σ and the matrix $\mathbf{A}$. The details of those two procedures are presented in [17]. After the depletion of the total budget, we use

B	Total budget (amount of resources) for tuning QPU annealing offsets (in units of total number of samples drawn from the QPU; In total, 10,000 samples are used in our experiment per instance.).
`InitialOffsets`	The initial value for each offset of qubits in the problem; we test either all set to 0 or uniformly between their min/max range.
`StepCost`	The cost of each step of the fitness evaluation of the offsets (in number of samples from the QPU; we used 100 samples per call).
$\mathbf{x}$	The current annealing offsets.
$f(\mathbf{x})$	Fitness value of the current offsets, measured in units of mean energy of the samples returned by the QPU.
c	Counter for the budget (measured in number of samples from the QPU).
CALL-QPU	The objective function that calls the QPU, takes $\mathbf{x}$ and `StepCost` as arguments.
σ	The step-size that scales the mutation of offsets.
$\mathbf{C}$	The matrix of covariances between the annealing offset values
$\mathbf{A}$	The Cholesky decomposition of $\mathbf{C}$
UPDATE-STEP-SIZE	The procedure to control the step-size σ. Please see [17] for the detail.
UPDATE-CHOLESKY	The procedure to adapt the Cholesky decomposition $\mathbf{A}$ of the covariance matrix $\mathbf{C}$. Please see [17] for details.
$\mathcal{U}(a, b)$	Uniform random distribution in $[a, b]$.
$\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$	Standard multivariate normal distribution.

Table 1. Explanation of variables and procedure used in Alg. 1.

the final annealing offsets produced by CMA-ES to solve the problem instances: $10,000$ samples are assigned to the QPU and the final annealing offsets $\mathbf{x}$ from Alg. 1 are used. Given a budget of $20,000$ samples per MIS instance, this means that 50% of the budget (per MIS instance) is allocated for offset tuning while the remaining 50% are used for problem solving. The goal is to allocate a sufficient amount of resources for calibration, and then solve the MIS problems with the remaining QPU time.

6 Experimental results

Similar to previous work [7], 50 random graphs with 20-60 variables were generated using edge probability $p = 0.2$, and the MIS problem was solved for each. The

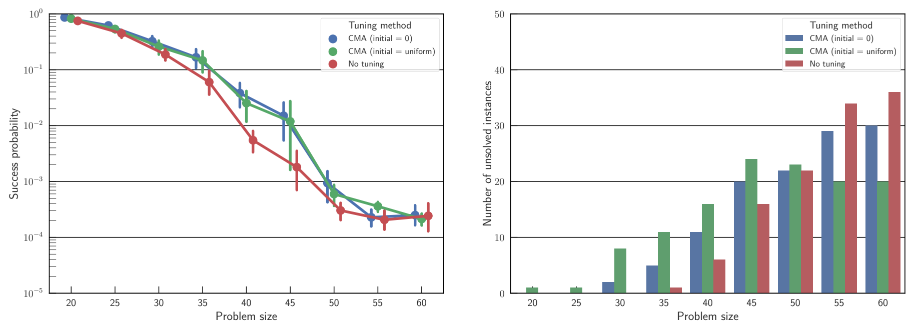

Fig. 1. Left: Mean success probability (of instances where tuned solvers found the optimum) for increasing problem sizes. Higher success probabilities are better, indicating improved performance in finding optima. **Right:** Bar chart showing how many problems remained unsolved by the solvers, before/after tuning with different configurations. As problem sizes increase, the no tuning version of the QPU solves relatively fewer instances.

QPU annealing offsets were tuned using 10,000 samples per instance as described in Section 5. Each tuned annealing offset configuration was then used to collect 10,000 samples from the QPU for each MIS instance, as well as the "no offsets" configuration for comparison. Figure 1 (left) shows the mean probability of success for configurations that found the presumed optimum. We show the two configurations of tuning (without initial offsets and uniform offsets) relative to the naive performance of no tuning. It is possible for certain configurations to not find the global optimum, even after tuning, as shown in Figure 1 (right). As expected, using the QPU with no annealing offsets leads to the highest amount of unsolved instances at the largest problem sizes. This means that, on average, the tuned annealing offsets find optima (larger independent sets) that *cannot be obtained without tuning*. Surprisingly, however, there are problem instances that remain unsolved even for smaller problem sizes with tuning. Additionally, the initial point for the CMA-ES routine (i.e., uniform vs. null initial offsets) affects the number of unsolved problems at smaller sizes.

The null initial offsets, which can be viewed as a stable local minimum for smaller problems, typically solve more instances than the uniform initial offsets configuration, although both are outperformed by no offsets (Figure 1, right). This is consistent with previous observations [7] where the QPU with no offsets was able to outperform even simulated thermal annealing at small problem sizes. However, in the cases where both tuning configurations found the optimum, Figure 1 shows that the probability of obtaining ground states is similar in both configurations, and both versions outperformed the QPU without tuning for all problem sizes. In Figure 2 we show the ratio of the mean success probability of instances where the ground state was obtained between the tuned and untuned offsets. This measures the improvement in probability of obtaining the ground state for a particular MIS instance. The improvement obtained by using the two configurations is qualitatively similar, and peaks at 12.4 times improvement at problem size 45 for the null initial offsets, and a factor of 8.2 improvement at problem size 45 for the uniform initial offsets. The improvement gained by tuning the annealing offsets steadily increases with problem size, until reaching its peak, after which the gains mostly disappear (although we are still able to solve a higher number of problems after tuning). This behavior indicates that at small problem sizes there is little to be gained from tuning, but at larger problem sizes the annealing offsets can have a significant impact on performance. The decay observed in success probability in problem sizes larger than 45 implies that insufficient resources were allocated to the tuning procedure, and more than 10,000 samples are needed to tune the offsets. Deriving such an optimal allocation of resources is beyond the scope of this paper. Additional analysis of the final offsets are shown in Appendix B. In real-world applications, it is impractical to exhaustively tune hyperparameters when using heuristics. Typically, either rules-of-thumb or iterative tuning should be used in order to work in practical timescales. In tuning annealing offsets for the QPU, the existing previous works are two papers from D-Wave Systems that employ two algorithms to change the annealing offsets [11, 12]. In each paper, different magnitudes of samples

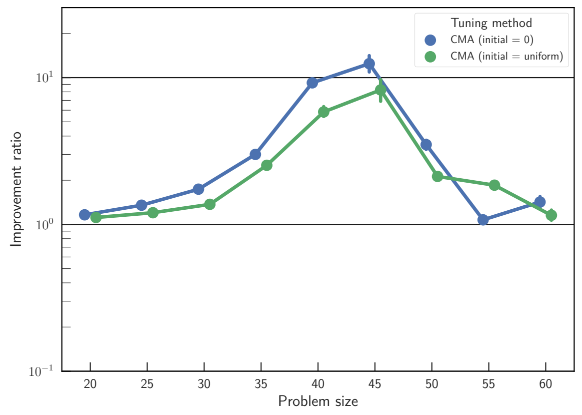

Fig. 2. Ratio of mean success probabilities before and after tuning, with different tuner configurations. Means are calculated only using cases where ground states were obtained (as per Figure 1), and errors were extracted via bootstrapping using 100 data points. Points above the line 10^0 indicate improvement by using the respective tuning configuration. Peak improvement is observed for problem size 45 with both configurations, with a mean improvement of 12.4 for the null initial offset configuration and a factor of 8.2 for the uniform configuration.

and success probability improvements are observed. In order to perform a fair comparison between the various methods, we introduce a performance factor calculated by dividing the number of samples used in the tuning procedure by the improvement factor obtained (the ratio of success probabilities before/after tuning). This performance factor can be interpreted as a measure of resource efficiency, and lower performance factor is better. The best improvement ratio obtained using the CMA-ES procedure introduced in this paper was 12.4, for the configuration of null initial offsets and problem size 45. This yields a factor of 806, which is more than 3 times lower than in the previous best in existing literature. We therefore find our method to be 3 times more efficient than the existing method at best. We also note that both versions of our tuning procedure were more resource efficient (better performance factor) than other methods. A full comparison is shown in Tab. 2.

Method	**Samples**	**Improvement**	**Performance**
D-Wave (grid)	$2.5 \cdot 10^6$	1000	2500
D-Wave (perturb.)	$3.15 \cdot 10^5$	1.37	$2.3 \cdot 10^4$
CMA-ES (uniform)	10^4	8.2	1219
CMA-ES (null)	10^4	12.4	**806**

Table 2. Table comparing the number of samples used, the success probability improvement ratio, and overall performance factor between the existing annealing offset tuning methods.

7 Conclusions

In this paper we introduced a novel method for heuristically tuning hyperparameters in existing quantum annealing processors. Using a (1+1)-CMA-ES algorithm to tune annealing offsets, we demonstrate an improvement of up to 12.4 times in probability of obtaining optima, and are able to find better solutions than without tuning. We are able to do this in a model-agnostic way that does not require domain-specific knowledge other than the bounds of the parameters. Additionally, we are able to make efficient use of our QPU samples, and are more than 3 times more efficient than the existing tuning techniques shown in [11, 12]. Improvements via tuning were obtained by exploring only a single parameter, the annealing offset parameter. Our results show that it is possible to use classical algorithms to iteratively tune hyperparameters and boost performance of commercially available QPUs, shown here on a test set of MIS problems. This result opens the door to new use cases for classical optimizers, and introduces a new paradigm for hybrid quantum/classical computing. In future work, we will investigate additional tuning algorithms, incorporate more parameters in the tuning process, and use additional problem sets to test the results.

References

1. J. King, S. Yarkoni, M. M. Nevisi, J. P. Hilton, and C. C. McGeoch, “Benchmarking a quantum annealing processor with the time-to-target metric.” arXiv:1508.05087, 2015.

2. Z. Bian, F. Chudak, R. B. Israel, B. Lackey, W. G. Macready, and A. Roy, “Mapping constrained optimization problems to quantum annealing with application to fault diagnosis,” *Frontiers in ICT*, vol. 3, p. 14, 2016.
3. F. Neukart, G. Compostella, C. Seidel, D. von Dollen, S. Yarkoni, and B. Parney, “Traffic flow optimization using a quantum annealer,” *Frontiers in ICT*, vol. 4, p. 29, 2017.
4. J. Raymond, S. Yarkoni, and E. Andriyash, “Global warming: Temperature estimation in annealers,” *Frontiers in ICT*, vol. 3, p. 23, 2016.
5. D. Venturelli, D. J. J. Marchand, and G. Rojo, “Quantum annealing implementation of job-shop scheduling.” arXiv:1506.08479, 2015.
6. A. D. King, J. Carrasquilla, J. Raymond, I. Ozfidan, E. Andriyash, A. Berkley, M. Reis, T. Lanting, R. Harris, F. Altomare, K. Boothby, P. I. Bunyk, C. Enderud, A. Fréchette, E. Hoskinson, N. Ladizinsky, T. Oh, G. Poulin-Lamarre, C. Rich, Y. Sato, A. Y. Smirnov, L. J. Swenson, M. H. Volkmann, J. Whittaker, J. Yao, E. Ladizinsky, M. W. Johnson, J. Hilton, and M. H. Amin, “Observation of topological phenomena in a programmable lattice of 1,800 qubits,” *Nature*, vol. 560, no. 7719, pp. 456–460, 2018.
7. S. Yarkoni, A. Plaat, and T. Bäck, “First results solving arbitrarily structured maximum independent set problems using quantum annealing,” in *2018 IEEE Congress on Evolutionary Computation (CEC)*, (Rio de Janeiro, Brazil), pp. 1184–1190, 2018.
8. M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, “Quantum annealing with manufactured spins,” *Nature*, vol. 473, pp. 194–198, May 2011.
9. F. Barahona, “On the computational complexity of ising spin glass models,” *Journal of Physics A: Mathematical and General*, vol. 15, no. 10, p. 3241, 1982.
10. A. Lucas, “Ising formulations of many NP problems,” *Frontiers in Physics*, vol. 2, p. 5, 2014.
11. T. Lanting, A. D. King, B. Evert, and E. Hoskinson, “Experimental demonstration of perturbative anticrossing mitigation using non-uniform driver hamiltonians.” arXiv:1708.03049, 2017.
12. E. Andriyash, Z. Bian, F. Chudak, M. Drew-Brook, A. D. King, W. G. Macready, and A. Roy, “Boosting integer factoring performance via quantum annealing offsets.” https://www.dwavesys.com/resources/publications.
13. T.-J. Hsu, F. Jin, C. Seidel, F. Neukart, H. D. Raedt, and K. Michielsen, “Quantum annealing with anneal path control: application to 2-sat problems with known energy landscapes.” arXiv:1810.00194, 2018.
14. Y. Susa, Y. Yamashiro, M. Yamamoto, and H. Nishimori, “Exponential speedup of quantum annealing by inhomogeneous driving of the transverse field,” *Journal of the Physical Society of Japan*, vol. 87, no. 2, p. 023002, 2018.
15. T. Kadowaki and H. Nishimori, “Quantum annealing in the transverse ising model,” *Phys. Rev. E*, vol. 58, pp. 5355–5363, Nov 1998.
16. N. Hansen, “The CMA Evolution Strategy: A Comparing Review,” in *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms* (J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, eds.), pp. 75–102, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

17. C. Igel, T. Suttorp, and N. Hansen, "A Computational Efficient Covariance Matrix Update and a (1+1)-CMA for Evolution Strategies," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, (New York, NY, USA), pp. 453–460, ACM, 2006.
18. A. Auger and N. Hansen, "Benchmarking the (1+1)-CMA-ES on the BBOB-2009 Noisy Testbed," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, (New York, NY, USA), pp. 2467–2472, ACM, 2009.

A Evaluating the fitness function of $(1 + 1)$-CMA-ES using the QPU

Here we show an example of a single tuning run of $(1 + 1)$-CMA-ES for a 40 node graph, with the configuration of initial offsets set to all zeroes. As explained in Alg. 1, we use the mean energy of 100 samples returned by the QPU as the evaluation for the fitness function of the CMA-ES. The sample size of 100 was determined empirically as being the minimum number of samples to determine the mean energy, and is consistent with previous results [4]. In Figure 3 (left) we show the progression of the CMA-ES routine and the associated fitness function. The tuning shows a clear improvement in mean energy, as shown both in the fitness function and the cumulative minimum of the fitness function. Every time the objective function improves, the respective annealing offsets that were used in that sample set are recorded. The evolution of the annealing offsets for this 40 variable instance is shown in Figure 3 (right). The final offsets after tuning were then used to test their performance.

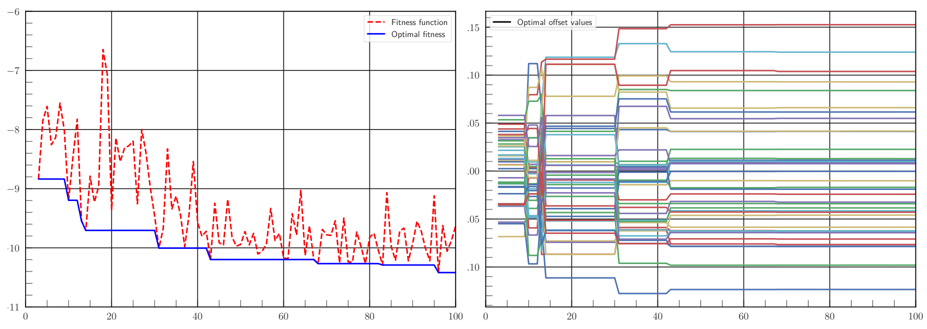

Fig. 3. Left: The fitness function evolution (mean energy of 100 samples) is shown as a function of the iteration number in CMA-ES. The red line represents the value of the fitness function at each iteration of CMA-ES, and the blue line is the cumulative minimum, representing the best solutions so far. **Right**: The evolution of the annealing offsets are shown as a function of the iteration number of CMA-ES (updated every time improvement is found by the CMA-ES).

B Analysis of tuned annealing offsets

Here we present the aggregated results of all the annealing offsets post tuning. Figures 4 (left and right) show the final offset values for all problem instances using the CMA-ES routine with initial offsets set to zero and uniform, respectively. We found that the final offset values were not correlated with chain length or success probability. However, we did see a systematic shift in the final offset values with respect to the degree of the node in the graph, and as a function of problem size. In both figures, we see divergent behavior in offsets for very small and very high degree nodes, with consistent stability in the mid-range. The main different between the two figures is the final value of the offsets in this middle region. In Figure 4 (left), the average offset value rises from 0 at small problems, to roughly .02 for problems with 40 variables, then back down to 0 for the largest problems. There is also a slight increase in average offset value from degree 3 to degree 14, found consistently for all problem sizes. In contrast, Figure 4 (right) shows that the final offset values were roughly .02 at all problem sizes, apart from the divergent behavior in the extrema of the degree axis. The difference between the two configurations could explain why initial offsets set to zero performed slightly better than the uniform initial offsets. Given the fixed resources of 10,000 samples for calibration per MIS instance, escaping from a local optimum (such as the null initial configuration) becomes increasingly difficult at larger problem sizes, thus degrading the uniform configuration's performance. Other than the results shown here, we were not able to extract any meaningful information with respect to other interesting parameters.

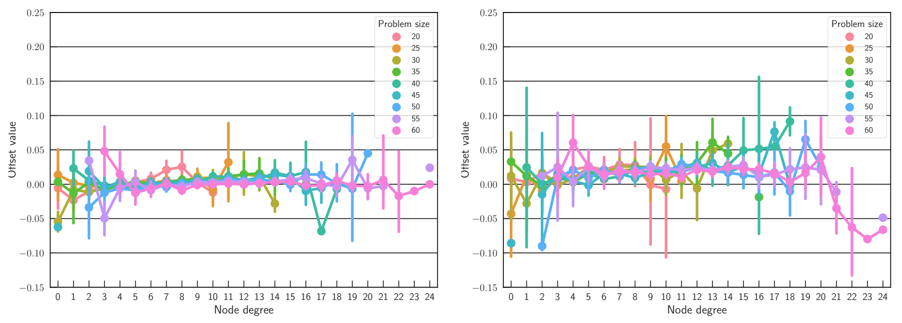

Fig. 4. Left: Final offset value as determined by (1+1)-CMA-ES with initial offsets set to zero, as a function of the degree of the logical node in the graph. Colors represent different problem sizes. **Right**: Same as in left, but for initial offsets set uniformly in their allowed range.