Postprocessing Methods on D-Wave Systems

09-1105A-E

Notice and Disclaimer

D-Wave Systems Inc. ("D-Wave"), its subsidiaries and affiliates, makes commercially reasonable efforts to ensure that the information in this document is accurate and up to date, but errors may occur. NONE OF D-WAVE SYSTEMS INC., its subsidiaries and affiliates, OR ANY OF ITS RESPECTIVE DIRECTORS, EMPLOYEES, AGENTS, OR OTHER REPRESENTATIVES WILL BE LIABLE FOR DAMAGES, CLAIMS, EXPENSES OR OTHER COSTS (INCLUDING WITHOUT LIMITATION LEGAL FEES) ARISING OUT OF OR IN CONNECTION WITH THE USE OF THIS DOCUMENT OR ANY INFORMATION CONTAINED OR REFERRED TO IN IT. THIS IS A COMPREHENSIVE LIMITATION OF LIABILITY THAT APPLIES TO ALL DAMAGES OF ANY KIND, INCLUDING (WITHOUT LIMITATION) COMPENSATORY, DIRECT, INDIRECT, EXEMPLARY, PUNITIVE AND CONSEQUENTIAL DAMAGES, LOSS OF PROGRAMS OR DATA, INCOME OR PROFIT, LOSS OR DAMAGE TO PROPERTY, AND CLAIMS OF THIRD PARTIES.

D-Wave reserves the right to alter this document and other referenced documents without notice from time to time and at its sole discretion. D-Wave reserves its intellectual property rights in and to this document and its proprietary technology, including copyright, trademark rights, industrial design rights, and patent rights. D-Wave trademarks used herein include D-WAVE®, D-WAVE 2X™, D-WAVE 2000Q™, and the D-Wave logo (the "D-Wave Marks"). Other marks used in this document are the property of their respective owners. D-Wave does not grant any license, assignment, or other grant of interest in or to the copyright of this document, the D-Wave Marks, any other marks used in this document, or any other intellectual property rights used or referred to herein, except as D-Wave may expressly provide in a written agreement. This document may refer to other documents, including documents subject to the rights of third parties. Nothing in this document constitutes a grant by D-Wave of any license, assignment, or any other interest in the copyright or other intellectual property rights of such other documents. Any use of such other documents is subject to the rights of D-Wave and/or any applicable third parties in those documents.

All installation, service, support, and maintenance of and for the D-Wave System must be performed by qualified factory-trained D-Wave personnel. Do not move, repair, alter, modify, or change the D-Wave System. If the equipment is used in a manner not specified by D-Wave, the protection provided by the equipment may be impaired. Do not provide access to the customer site to anyone other than authorized and qualified personnel. Failure to follow these guidelines may result in disruption of service, extended downtime, damage to equipment (customer's, D-Wave's, and/or third parties'), injury, loss of life, or loss of property.

CONTENTS

CHAPTER

ONE

INTRODUCTION

The D-Wave™ system enables users to run postprocessing optimization and sampling algorithms on solutions obtained through the quantum processing unit (QPU). Postprocessing provides local improvements to these solutions with minimal overhead.

This document provides an overview of the postprocessing methods available, presents the results of a number of tests conducted by D-Wave, and describes the role of postprocessing in results obtained by the virtual full-yield Chimera (VFYC) solver.

CHAPTER

TWO

TYPES OF POSTPROCESSING

Solutions obtained from a D-Wave quantum processing unit (QPU) can be postprocessed to achieve higher quality.

When submitting a problem to the QPU, users choose from:

- No postprocessing (default)
- Optimization postprocessing
- Sampling postprocessing

For optimization problems, the goal is to find the state vector with the lowest energy. For sampling problems, the goal is to produce samples from a specific probability distribution. In both cases, a logical graph structure is defined and embedded into the QPU's Chimera topology. Postprocessing methods are applied to solutions defined on this logical graph structure.

The flow of integrated postprocessing is shown in Fig 2.1.

Optimization Postprocessing

The goal of optimization postprocessing is to obtain a set of samples with lowest energy on a graph G. For simplicity of discussion, assume that G is a logical graph. Pseudocode for optimization postprocessing is shown below. The post-processing method works by performing local updates to the state vector S based on low treewidth graphs. Specifically, the *DecomposeGraph* function uses a set of algorithms based on the minimum-degree *[Markowitz1957]* heuristic to decompose graph G into several low treewidth subgraphs that cover the nodes and edges of G. The *SolveSubgraph* function is then used to update the solution on each subgraph to obtain a locally optimal solution s'. *SolveSubgraph* is an exact solver for low treewidth graphs based on belief propagation on junction trees *[Jensen1990]*.

Data: Set S majority voted samples from the QPU, logical graph G
Result: Set S' post-processed results
$G' \leftarrow$ `DecomposeGraph(`G`)`
$S' \leftarrow \{\}$
for $s \in S$ **do**
  $s' = s$
  repeat
    $e \leftarrow E(s')$
    for $g \in G'$ **do**
      $s' \leftarrow$ `SolveSubgraph(`g, s'`)`
    end
  until $E(s') = e$;
  $S' \leftarrow S' \cup \{s'\}$
end

Algorithm 1: Optimization postprocessing algorithm

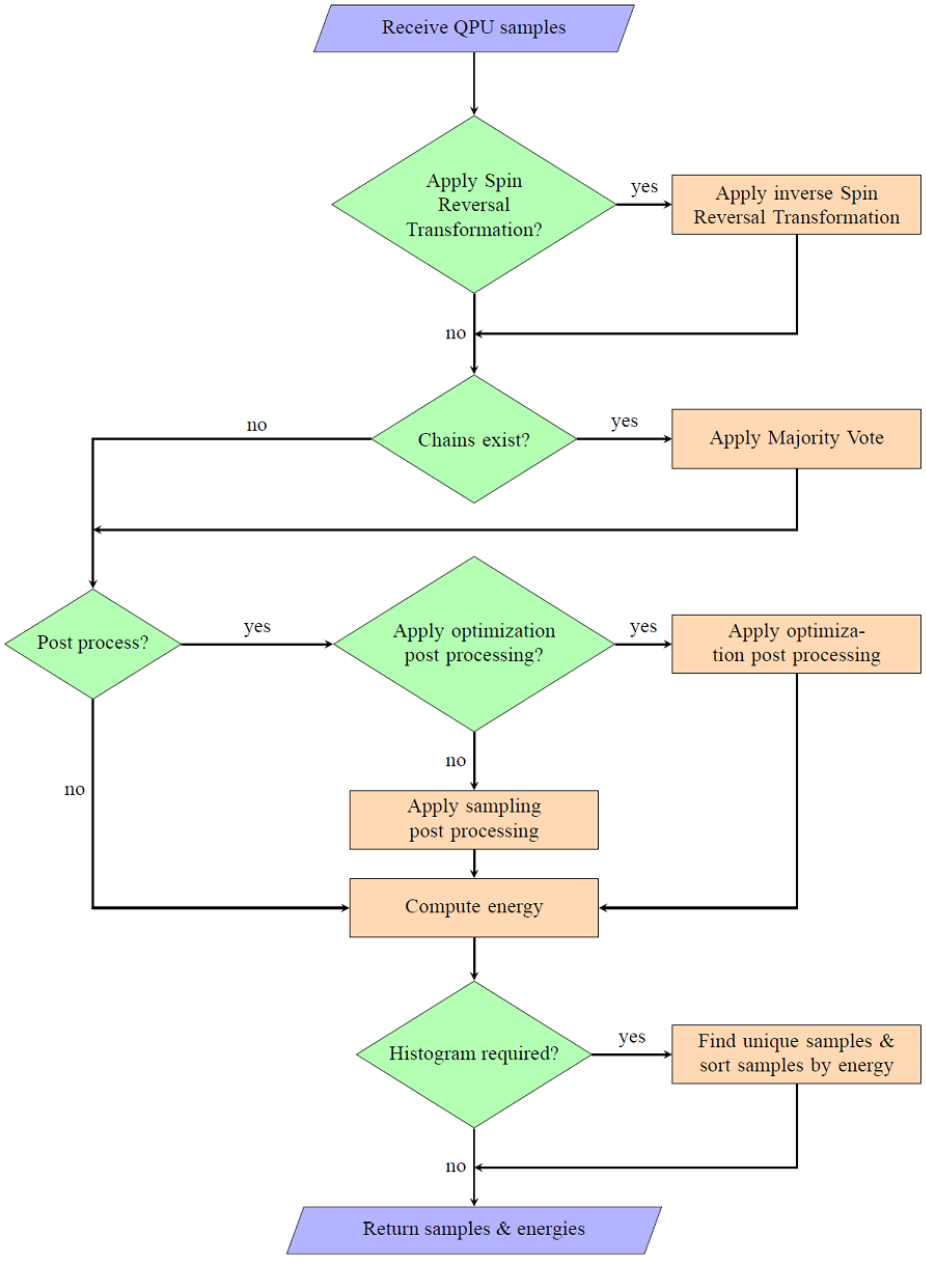

Fig. 2.1: Postprocessing flowchart

Sampling Postprocessing

In sampling mode, the goal of postprocessing is to obtain a set of samples that correspond to a target Boltzmann distribution with inverse temperature β defined on the logical graph G. The pseudocode for sampling postprocessing is shown below. As with optimization postprocessing, the graph G is decomposed into low treewidth subgraphs G' that cover G. On each subgraph, the state variables S obtained from the QPU are then updated using the *SampleSubgraph* function to match the exact Boltzmann distribution conditioned on the states of the variables outside the subgraph. Users can set the number of subgraph update iterations n; the choice of β is also determined by the user. Heuristically, this inverse temperature depends on model parameters in the Hamiltonian. As a general rule, β should be set to a value close to an inverse temperature corresponding to raw QPU samples. Some variation from a classical Boltzmann distribution is expected in postprocessed samples. See *Sampling Tests and Results* for discussion.

Data: Set of majority voted samples S from QPU, logical graph G, inverse temperature β, iteration number n
Result: Set S' post-processed results
$G' \leftarrow$ `DecomposeGraph(`G`)`
$S' \leftarrow \{\}$
for $s \in S$ **do**
  $s' = s$
  for $i \leftarrow 1$ **to** n **do**
    for $g \in G'$ **do**
      $s' \leftarrow$ `SampleSubgraph(`g, s', β`)`
    end
  end
  $S' \leftarrow S' \cup \{s'\}$
end

Algorithm 2: Sampling post-processing algorithm

CHAPTER

THREE

EXECUTION AND TIMING

The two main functions for the postprocessing algorithms are the low treewidth graph decomposition and the low treewidth graph solve. Graph decomposition takes place on the postprocessing server during the QPU access time for the previous problem, while the low treewidth solve is interwoven within the QPU duty cycle.

Fig 3.1 shows how a set of samples are batched and sent through the postprocessing solver as the next batch is being computed by the QPU. The total time spent postprocessing samples is provided in the timing structure as `total_post_processing_time`. Since all but the last batch are processed during the QPU duty cycle, the additional overhead time for postprocessing, `post_processing_overhead_time`, corresponds to the time spent processing the final batch.

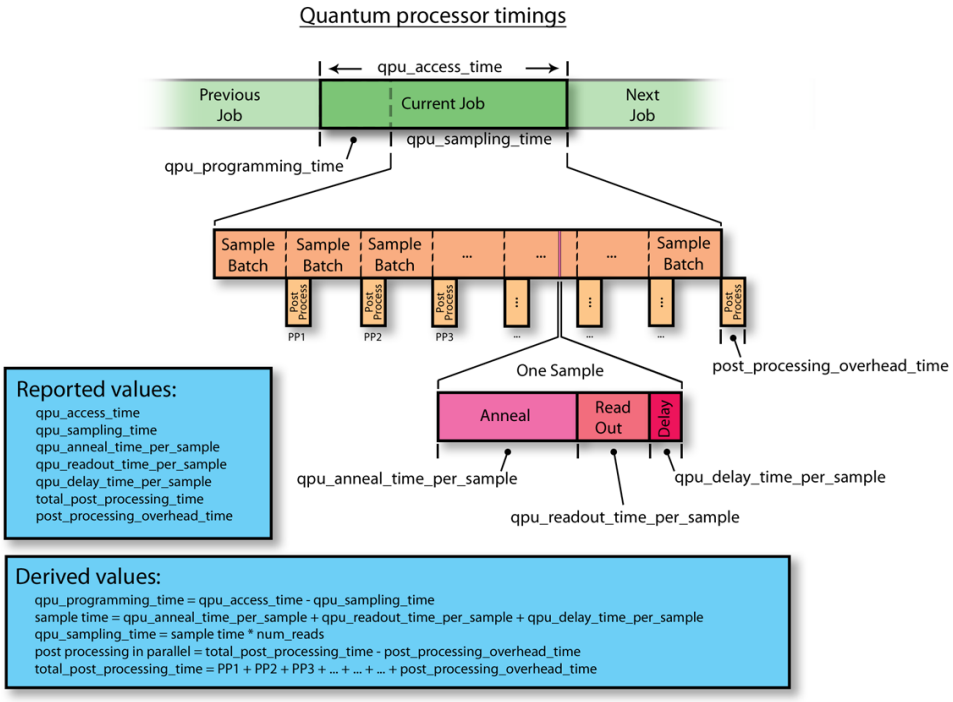

Fig. 3.1: D-Wave QPU timing structure

For more details about the timing structure, see *Measuring Computation Time on D-Wave Systems*, available on the Qubist web user interface.

CHAPTER

FOUR

OPTIMIZATION TESTS AND RESULTS

This section describes a set of tests and results that examine the quality of results returned by the optimization post-processor. The goal is to describe the difference between postprocessed solutions from those obtained solely via the QPU.

Postprocessing for optimization is evaluated by generating 10 random problems on a D-Wave QPU, each with J values drawn uniformly at random from $\{1, -1\}$. Each problem is evaluated based on a set of scaling factors. Problems are scaled to exaggerate the negative effects of analog noise on solution quality, so the optimization postprocessor can demonstrate that it can recover a high-quality solution from the QPU solution. Specifically, with small scaling factors, it is difficult to faithfully represent problems in the QPU because of the exaggeration of analog noise. This noise causes the solver to return lower-quality solutions, and provides a nice mechanism to evaluate the optimization postprocessor.

For each problem and scaling factor, 1000 samples were drawn with postprocessing on and off. As seen in Fig 4.1 and Fig 4.2, postprocessing for optimization can improve solutions significantly. Furthermore, the worse the non-postprocessed solutions are, the more postprocessing helps.

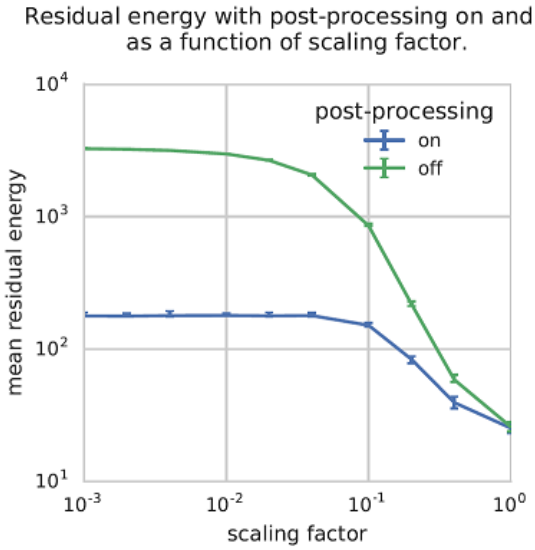

Fig. 4.1: Line plot of mean residual energies (mean energies above ground-state energy) returned by the D-Wave system with and without optimization postprocessing. Observe that optimization postprocessing does no harm, and helps more when scaling factors are smaller and the non-postprocessed samples not as good. Error bars indicate 95% confidence intervals over input Hamiltonians.

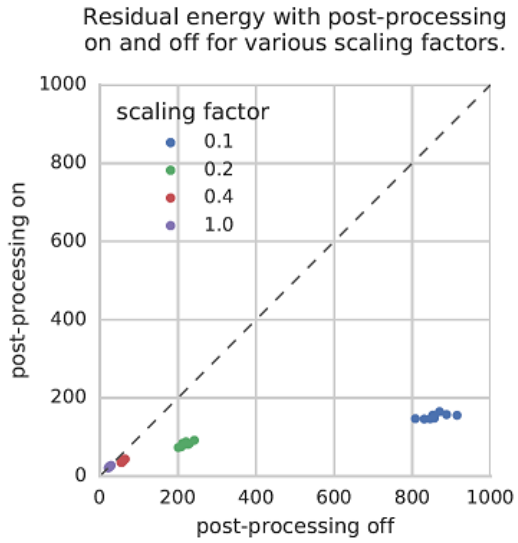

Fig. 4.2: Scatter plot of mean residual energies (mean energies above ground-state energy) returned by the D-Wave system with and without optimization postprocessing, with each point representing an input Hamiltonian. Observe that optimization postprocessing does no harm, and helps more when scaling factors are smaller and the non-postprocessed samples not as good.

CHAPTER

FIVE

SAMPLING TESTS AND RESULTS

This section describes tests conducted to examine the quality of results returned by the sampling postprocessor. The goal is to describe the difference between postprocessed samples from those obtained solely via the QPU. Postprocessing is considered here at two different temperatures: an ultra-cold temperature, and a measured local temperature *[Raymond2016]*.

The results show that the energy improves for cold temperature, but at the cost of diversity. For the local temperature, diversity of postprocessed samples improves without compromising the energy distribution. Measures such as false-positive rate and backbone, defined below, complement the typical energy/entropy statistics. The results show that the backbone and false-positive rates are improved by low-temperature postprocessing. Together, these studies provide a glimpse into the behavior of QPU and postprocessed samples.

Methodology

The study considers a particular problem class: Not-All-Equal-3SAT (NAE3SAT), with 30 logical variables and a clause-to-variable ratio of 1.8. Each clause is a logical constraint on three variables, and the energy of a sample is linear in the number of clause violations.[1] This class was chosen because a variety of meaningful metrics can be used to analyze the raw QPU samples and the postprocessed results *[Douglass2015]*. The embedding of this problem was chosen using the standard routine *[Cai2014]*, and chain strength for the embedding was chosen by a heuristic rule that gave close-to-optimal results in terms of the fraction of ground states seen without postprocessing.

Sample quality is evaluated with respect to a target Boltzmann distribution using two values of β: an ultra-cold temperature corresponding to $\beta = 10$ and a local estimate corresponding to $\beta = 2.0$. The cold temperature was chosen to be (for practical purposes) indistinguishable from the limiting case $\beta \to \infty$. In this limited postprocessing, samples can only decrease in energy. This is a useful limit when the objective is to obtain ground-state, or low-energy, samples. In the examples presented, a significant fraction of samples are ground states both in the raw QPU sample set and in the postprocessed sample set. The local temperature is chosen so that before and after postprocessing, the sample-set mean energy is approximately unchanged. The local temperature can be estimated accurately by an independent procedure that probes the average energy change as a function of β *[Raymond2016]*. Postprocessing at this local temperature should produce more diverse samples (higher entropy distributions) without increasing the average energy. This should be observed in our sampling metrics.

Fig 5.1 through Fig 5.5 show sample quality before and after postprocessing with $n = 10$, for various sampling metrics. Each pair of plots shows results from postprocessing at low temperature $\beta = 10$ (left) and local temperature $\beta = 2$ (right). Each panel shows results from 100 random NAE3SAT instances generated on 30 variables with clause-to-variable ratio 1.8. For each input, 1000 samples were collected from 10 different spin-reversal transforms for a total of 10,000 samples. The default annealing time of $20\mu s$ was used for all samples, and the postprocessor was applied with $n = 10$ steps. QPU samples have undergone embedding and chain-correction, and the following analysis is performed entirely in the logical space. Depending on the test being performed, sometimes only a subset of the samples were used. For example, it is convenient to define some metrics with respect to ground states only, or randomly select pairs

[1] The actual energy penalty for a clause violation is model dependent since the Hamiltonian is scaled to meet the QPU constraint on coupling strength.

of solutions to examine. Standard errors, and estimator bias (where relevant), are evaluated with the jackknife method *[Efron1982]*.

Mean Energy

Fig 5.1 demonstrates the mean energy for solutions to the test problems compared before and after postprocessing. The mean energy is the expectation of the sample set energies; this estimator is unbiased and of small variance.

If we postprocess at low temperature, we hope to transform excited states into low-energy ones, so that we aim for a decrease in mean energy under postprocessing. In the cold case, shown on the left, the mean energy decreases dramatically after postprocessing, which is suggestive of successful postprocessing.

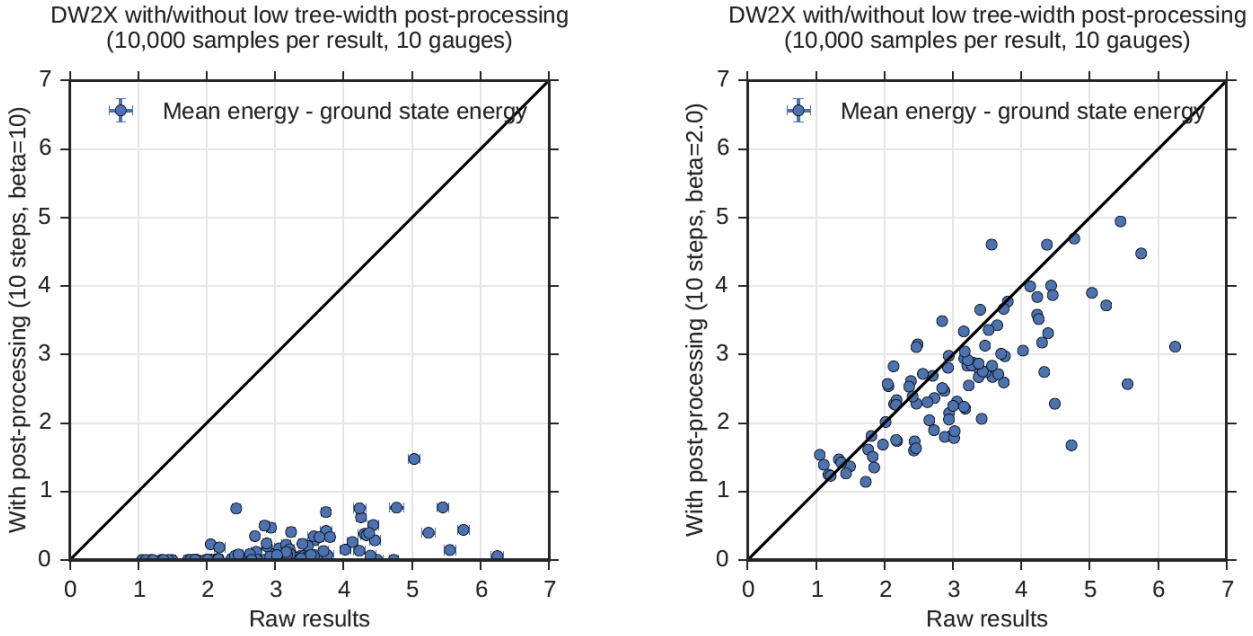

Fig. 5.1: Mean energy comparison of solutions to 100 NAE3SAT problems before and after postprocessing. Postprocessing is performed at $\beta = 10$ (left) and $\beta = 2$ (right). Postprocessing at $\beta = 10$ significantly reduces the energy of samples, whereas postprocessing at $\beta = 2$ does not. Standard errors are shown for each estimate, but these are in most cases small compared to the marker size.

If we postprocess at some other temperature, our aim is to approach the mean energy of the Boltzmann distribution at β. We have chosen our local temperature so that to a first approximation energy should be unchanged under postprocessing. However, we have chosen a single value of β for all NAE3SAT problems, so that we may expect some upward or downward fluctuation in mean energy in any given problem. Fig 5.1 (right) shows that, despite some fluctuations between problem instances, the mean energies before and after are, in the typical case, strongly correlated. This suggests only that our approximation to β local was appropriate for this class.

Entropy

Entropy is a measure of the size of the solution space. The aim of postprocessing at low temperature is to approach the ground-state entropy (a uniform distribution over ground states); in this process, the sample diversity is typically reduced. Successful postprocessing at our local β — chosen so that energy is approximately constant — leads to

an increase in entropy. The Boltzmann distribution is the maximum entropy distribution for a given mean energy; therefore, if mean energy is unchanged, we expect to see the largest value for entropy in the Boltzmann distribution.

The entropy for a distribution $P(x)$ is defined as $-\sum_x P(x) \log P(x)$, and can be estimated using samples drawn from P. The Grassberger estimator *[Grassberger2008]* was used to measure the entropy from our sample sets. Fig 5.2 shows the relative entropy of the samples before and after postprocessing. At the cold temperature, the entropy decreases significantly, likely due to many of the excited states returned by the QPU being transformed into ground states. This also follows from the mean energy plot in Fig 5.1. At local β, the entropy increases as one would expect. Low treewidth postprocessing allows the samples to diversify toward the maximum entropy distribution. This later choice of β allows a fair comparison of the two distributions since we control for the mean energy; otherwise, entropy can always be improved by raising the mean energy of the distribution.

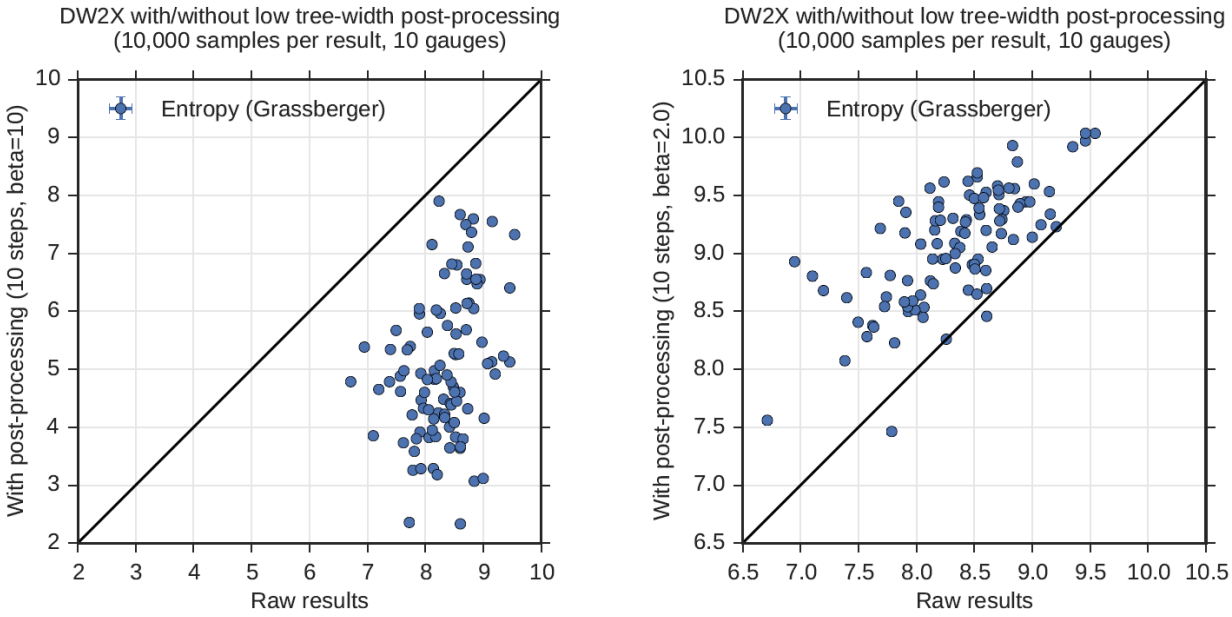

Fig. 5.2: Entropy comparison of solutions to 100 NAE3SAT problems before and after postprocessing. Postprocessing is performed at $\beta = 10$ (left) and $\beta = 2$ (right). Postprocessing at $\beta = 10$ reduces the entropy whereas postprocessing at $\beta = 2$ increases it.

KL Divergence

The Kullback–Leibler (KL) divergence is defined as β Energy $-$ Entropy $+\log(Z(\beta))$, where $Z(\beta)$ is a constant called the partition function. It is an important measure of distributional distance and is bounded below by zero. Postprocessing typically has a trade-off between mean energy and entropy. Distributions of high diversity (e.g., random samples) typically have higher energy; KL divergence is able to capture the trade-off between decreasing mean energy and increasing entropy. For any β, as a distribution approaches the Boltzmann distribution, its KL divergence decreases toward zero. Postprocessing as undertaken here is guaranteed to decrease KL divergence. The more successful the postprocessing is, the larger the decrease, and the closer the postprocessed distribution is to zero.

To demonstrate the effectiveness of postprocessing, we need not know the constant $\log(Z)$; we present instead $KLD' = (KLD - log(Z))/\beta$. Fig 5.3 shows a significant and consistent decrease in KL divergence for all cases. In the cold case, the improvement in KL divergence is largely due to decreases in the mean energy. For the local temperature postprocessing, the decrease is a result of increased sample diversity.

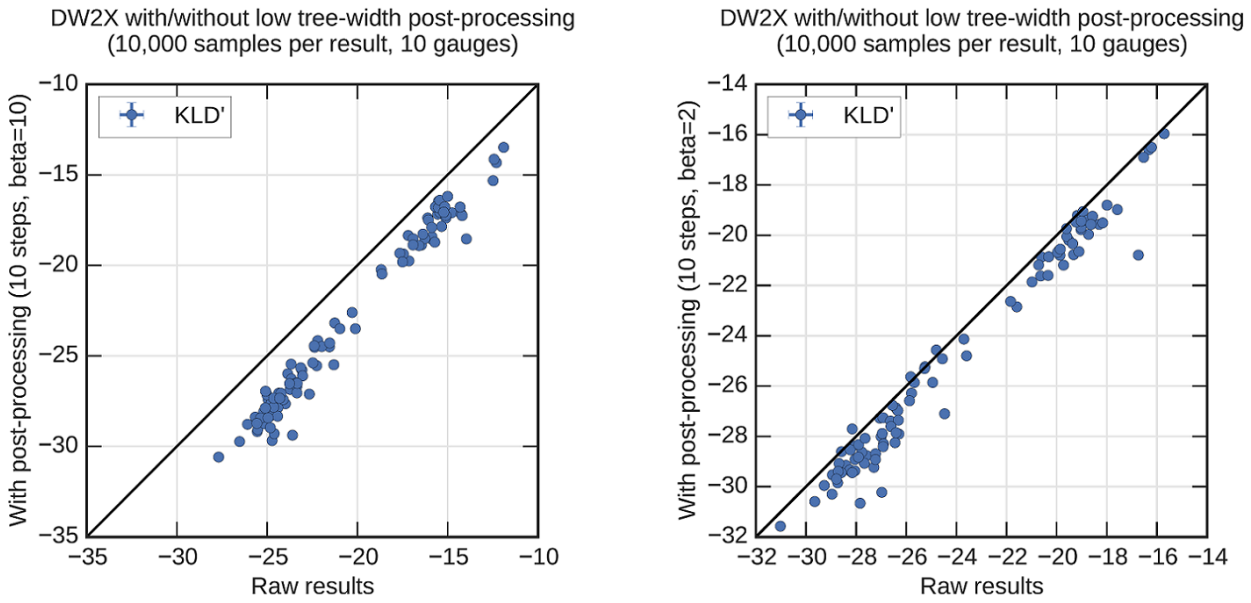

Fig. 5.3: KLD' comparison of solutions to 100 NAE3SAT problems before and after postprocessing. Postprocessing is performed at $\beta = 10$ (left) and $\beta = 2$ (right). In both cases, postprocessing improves the KL divergence, though the improvement is more significant at $\beta = 10$.

Backbone

If we restrict attention to the subset of samples that are ground states, we can define the backbone as the set of all variables that are perfectly correlated. Since the problem is symmetric, it is most interesting to consider edge states rather than spin states; to consider the number of correlations among variable pairs that are perfect (either perfect alignment or antialignment). The measure is interesting only for problems with ground-state degeneracy, such as NAE3SAT.

We define the backbone over edge variables for a distribution P as

$$b = \frac{1}{|E|} \sum_{(i,\, j) \in E}^{E} I\left(|\langle x_i x_j \rangle_{GS}| \equiv 1\right), \tag{5.1}$$

where E is the set of edges in the problem, x_i is the spin state of variable i, and angled brackets indicate an average with respect to the distribution over ground states (for the energy function of interest). $I()$ is an indicator function, evaluating to 1 when the argument is true. For a distribution that has nonzero probability to see all ground states, the backbone is equal to that of the Boltzmann distribution[2]. If the distribution covers only a subset of ground states, the backbone is larger than the Boltzmann distribution. In the special case that only a single ground state is observed, the maximum value (1) is obtained.

We estimate the backbone from a finite set of samples drawn from P by replacing $\langle x_i x_j \rangle$ with an empirical estimate[3]. This estimator is sensitive to not only whether the distribution supports a given ground state, but also how frequently the ground state is sampled. For a finite number of samples, the backbone is minimized by a distribution that is well spread across the ground state[4]. Consider the special case of a sampler that mostly sees a single ground state. If we only draw a few samples, the backbone is estimated as 1, even if by drawing many samples we would see the correct result. By contrast, if many diverse samples are drawn, a much smaller value is found. The Boltzmann distribution is

[2] All Boltzmann distributions see the ground states uniformly, though the probability to see a ground state is not uniform and increases with β.

[3] We draw this finite set from the empirical set restricting to ground states and without replacement. This ensures that the estimate is independent from the fraction of samples that are ground states, which may vary between distributions.

[4] A uniform distribution is expected to be close to optimal for many Hamiltonians, though it is not optimal in general.

uniform on the ground states and has a small backbone. Effective postprocessing reduces the estimate of the backbone as sample diversity increases and the Boltzmann distribution is approached.

Fig 5.4 shows the expected backbone when subsampling only two samples from the distribution. After postprocessing is applied with the cold temperature, the average backbone estimate produced by the sample improves overall. The trend is similar but less pronounced at the local temperature.

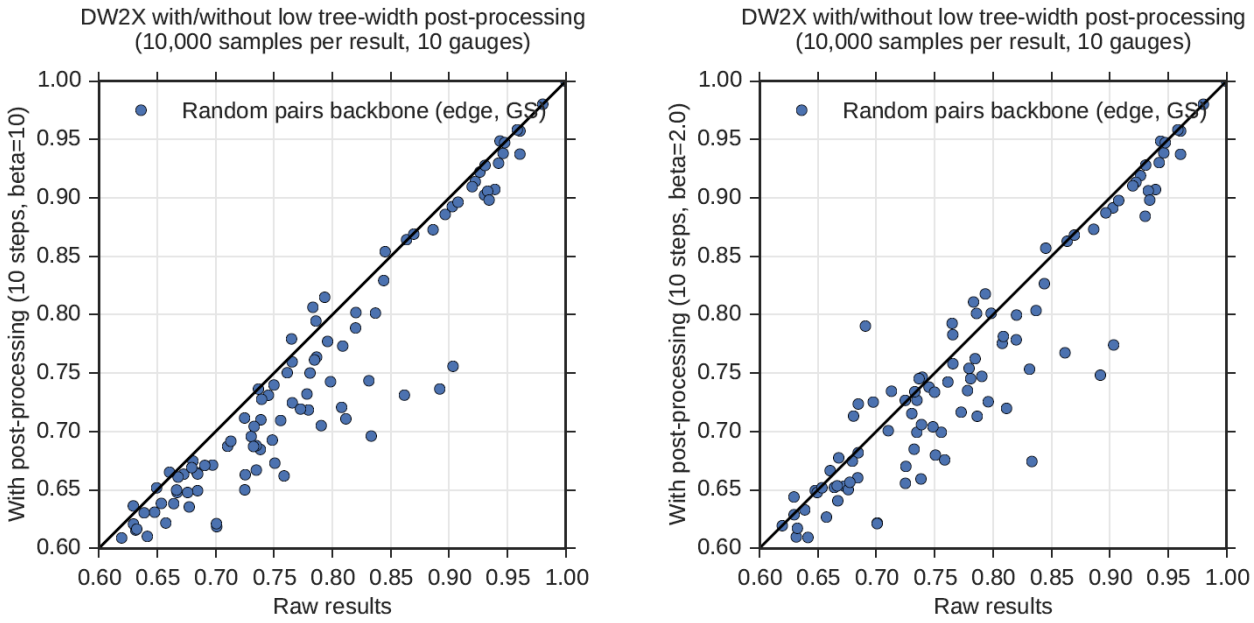

Fig. 5.4: Backbone comparison of solutions to 100 NAE3SAT problems before and after postprocessing. Postprocessing is performed at $\beta = 10$ (left) and $\beta = 2$ (right). In both cases, postprocessing improves the average backbone estimate overall, though the improvement is more significant at $\beta = 10$.

False-Positive Rate

One application of sampling is to create SAT filters *[Douglass2015]*. We present only a brief abstract description: the filter is defined by a set of samples, $\mathcal{S} = \{x\}$, which are ground-state solutions to a satisfiable NAE3SAT problem instance. A test T of this filter is defined by a random triplet of indices, i_1, i_2, i_3, and negations, n_1, n_2, n_3. Indices are sampled uniformly from 1 to N ($N = 30$, the number of logical variables) without repetition; negations, from ± 1. The test yields either a false positive (1) if every sample in the set passes the test

$$\wedge_{x \in \mathcal{S}} I \left(\left| \sum_{l=1}^{3} n_l x_{i_l} \right| \equiv 1 \right)$$

or zero otherwise. The false-positive rate (FPR) for a given filter is an average over tests; the FPR for a distribution is an average over tests and sample sets of a given size.

Including a diverse set of samples (ground states) in the filter yields a lower FPR, for much the same reason as it reduces the backbone. This reduction in the FPR relates directly to filter effectiveness in application. Thus, postprocessing can be deemed successful if the set of ground states produced are diversified — yielding lower FPRs.

Fig 5.5 demonstrates the performance of filters. The FPR is determined by a double average over tests and sample sets. Filters were constructed from 1000 random sample pairs drawn without replacement from the larger sample set; to each was applied 1000 random tests. Postprocessing improves the FPR in the majority of cases, though the signal

is not very strong. Trends in this metric are consistent with the backbone result, as would be expected because the backbone can be considered to place a limit on the FPR.

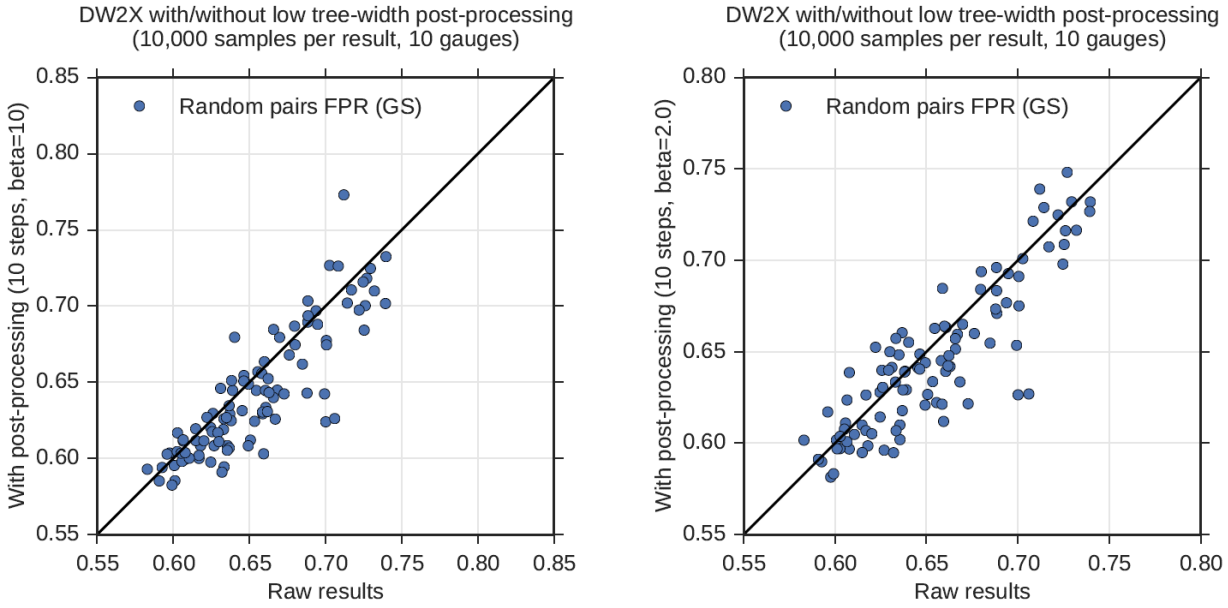

Fig. 5.5: Relative FPR comparison of solutions to 100 NAE3SAT problems before and after postprocessing. Postprocessing is performed at at $\beta = 10$ (left) and $\beta = 2$ (right). In both cases, postprocessing improves FPR overall, though the improvement is more significant at $\beta = 10$.

CHAPTER

SIX

VIRTUAL FULL-YIELD CHIMERA SOLVER

Each D-Wave system has a quantum processing unit (QPU) based on Chimera topology with a specific working graph that depends on the qubit yield. System solvers are configured to enable problems to be solved on the corresponding QPU working graph. Variations between working graphs require software and algorithms to be explicitly ported between systems. Developers who want to build portable software and prototype algorithms on an idealized processor abstraction need the ability to solve problems on a virtual full-yield Chimera (VFYC) graph. Using this abstraction, problems can be posed based on the Chimera topology rather than a specific working graph.

The VFYC solver provides such an interface to the system. Through this solver, variables corresponding to a Chimera structured graph that are not representable on a specific working graph are determined via hybrid use of the QPU and the integrated postprocessing system. As well for problems explicitly defined on the Chimera topology, this solver can be used to drive solutions implicitly on embedded problems. Preliminary studies show that solver performance depends on the qubit yield of the specific QPU, which varies system to system. Preliminary test results are discussed below.

The VFYC workflow is as follows:

1. A user submits a problem based on a full-yield Chimera graph with default postprocessing parameters.
2. The QPU solves the portion of the problem that corresponds to the QPU working graph.
3. Inverse spin-reversal transforms are applied.
4. The state of variables that are not represented via the QPU working graph are determined using the *SolveSubgraph* function shown in the *Optimization_Postprocessing* section. In the context of the VFYC solver, the specific subgraph g is chosen as the set of unrepresented variables, and the initial state s is determined from the solutions obtained in Step 3.
5. The final solutions are obtained by sending the states from Step 4 to initialize postprocessing once more in either sampling or optimization mode across the VFYC graph.

The following pseudocode shows the VFYC algorithm.

Data: Set S samples from the QPU, logical graph G, QPU working graph G_w
Result: Set S' samples for the virtual full yield Chimera
$S' \leftarrow \{S$ for variables in G_w, 0 for variables in $G - G_w\}$
for $s' \in S'$ **do**
  $s' \leftarrow$ `SolveSubgraph` $(G - G_w, s')$
end

Algorithm 3: Virtual full-yield Chimera algorithm

Performance Test Results

To test the performance of VFYC solvers, we compared the relative performance of a D-Wave QPU that had a set of missing qubits against a VFYC solver on the same underlying QPU with an additional 1% of the qubits disabled and

replaced via the hybrid VFYC solver.

We compared the two solvers on instances at various sizes from various problem classes, including RAN1 problems (random J values of ± 1), not-all-equal 3SAT (NAE3SAT) problems, and Google cluster-lattice (GCL) problems (see Denchev *et al. [Denchev2015]*).

For each problem class, the two solvers are compared in terms of mean residual energy and samples-to-solution (STS):

- Residual energy is defined as the energy above ground-state energy; in this case, the ground-state energy is estimated using exponential-time heuristic software solvers.
- STS is defined as the expected number of samples required to hit a ground state.

For all problem classes, the full solver performed better than the VFYC solver. For most problem classes, the VFYC solver performed reasonably well, and it performed particularly well on problems with chains, presumably because postprocessing in the logical solution space is more powerful than postprocessing in the native Chimera topology.

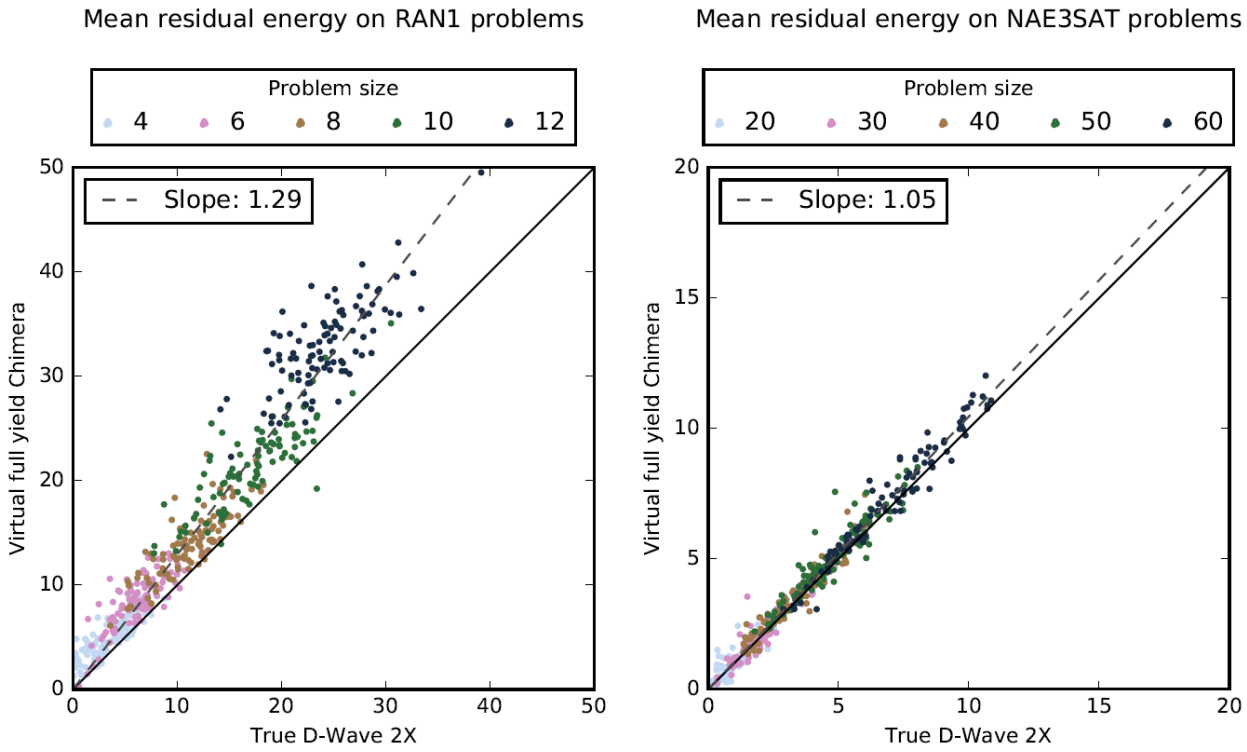

Fig. 6.1: Comparison of the mean residual energies yielded by a true D-Wave 2X solver that has a working graph Chimera topology and a VFYC solver on the same QPU, with 1% of the qubits disabled and repaired with postprocessing. Scatter plots show mean residual energies of the two solvers for problems of various sizes for RAN1 problems (left) and NAE3SAT problems (right). For RAN1 problems, the largest problem size shown runs on 12 x 12 unit cells. For NAE3SAT problems, the largest is a 60-variable problem. The performance of the VFYC QPU is inferior to that of the true D-Wave 2X QPU, but is still reasonably good. For NAE3SAT problems, the repaired solutions are almost as good as those returned by the fully intact QPU.

The one problem class in which the VFYC performed poorly was the GCL problem class. This is unsurprising because the strong performance that the D-Wave system demonstrates on these problems *[Denchev2015]* relies on coherent multiqubit tunneling. When missing qubits are simply ignored, the delicately balanced clusters in the GCL problems are badly misspecified, and postprocessing is unable to fix the problems. Put another way, GCL problems depend very strongly on the quantum dynamics used in computation within the D-Wave system, and the hybrid solver is an inadequate substitute.

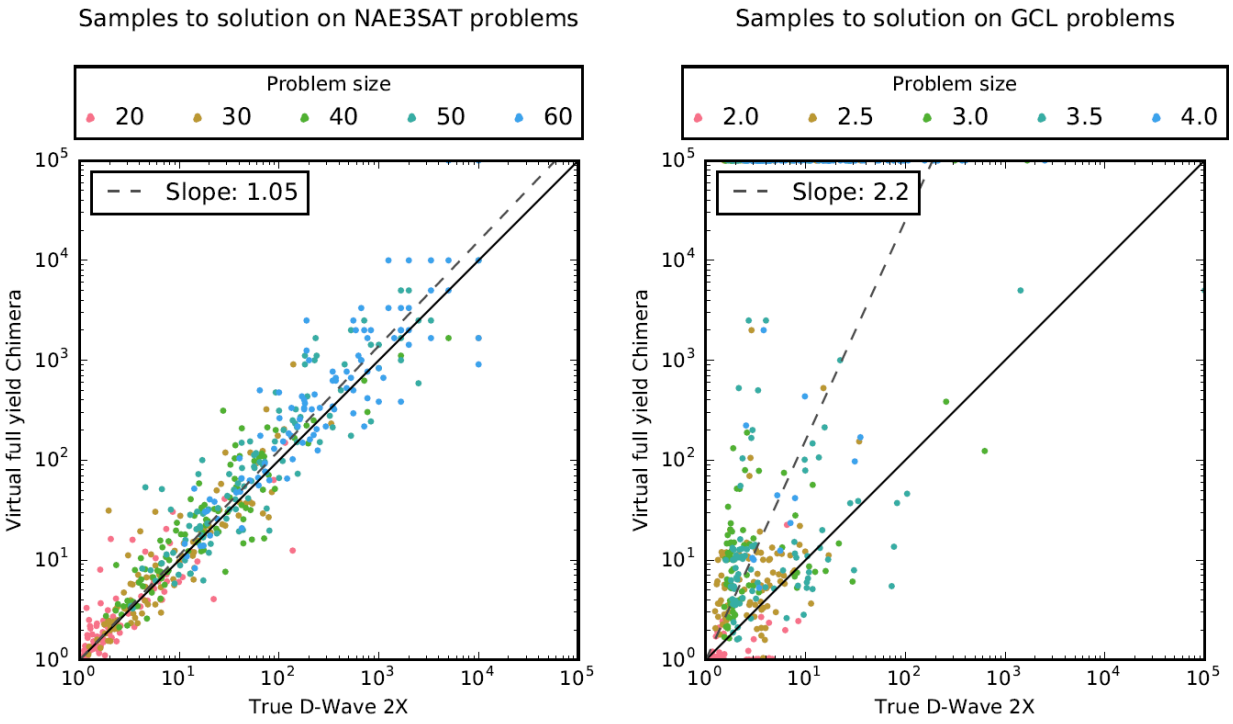

Fig. 6.2: Comparison of STS values yielded by a D-Wave 2X system that has a working graph Chimera topology and a VFYC solver on the same QPU, with 1% of the qubits disabled and repaired with postprocessing. Scatter plots show values of STS of the two solvers on problems of various sizes for NAE3SAT problems (left) and GCL problems (right). For NAE3SAT problems, the largest problem size shown is a 60-variable problem. For GCL problems, the largest runs on a 4 x 4 lattice of unit cells. The performance of the VFYC QPU is only slightly inferior to that of the true D-Wave 2X QPU for NAE3SAT problems. However, its performance is poor for GCL problems, suggesting that VFYC solvers should not be used for this class of problem.

BIBLIOGRAPHY

[Cai2014] J. Cai, B. Macready, and A. Roy, (2014). *A Practical Heuristic for Finding Graph Minors.* arXiv:1406.2741.

[Denchev2015] V.S. Denchev, S. Boixo, S.V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven (2015). *What is the Computational Value of Finite Range Tunneling?*, arXiv:1512.02206.

[Douglass2015] A. Douglass, A.D. King, and J. Raymond, (2015). *Constructing SAT Filters with a Quantum Annealer, Theory and Applications of Satisfiability Testing,* pp. 104–120. Springer International Publishing.

[Efron1982] B. Efron, (1982). *The Jackknife, the Bootstrap, and Other Resampling Plans.* Philadelphia, PA: Society for Industrial and Applied Mathematics. ISBN 9781611970319.

[Grassberger2008] P. Grassberger, (2008). *Entropy Estimates from Insufficient Samplings,* arXiv:physics/0307138v2.

[Jensen1990] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen (1990). *Bayesian updating in causal probabilistic networks by local computations,* Computational Statistics Quarterly, vol. 4, p. 269-282.

[Markowitz1957] H.M. Markowitz (1957). *The Elimination Form of the Inverse and its Application to Linear Programming,* Management Science, vol. 3, no. 3, p. 255-269.

[Raymond2016] J. Raymond, S. Yarkoni and E. Andriyash (2016). *Global Warming: Temperature Estimation in Annealers,* arXiv:1606.00919.