Experiments with QUBO on D-Wave

Viktória Nemkin*†

Department of Computer Science
and Information Theory
Budapest University of Technology
and Economics
nemkin@cs.bme.hu

Qusay Alghazali

Department of Telecommunications
and Artificial Intelligence
Budapest University of Technology
and Economics
qusay@tmit.bme.hu

Tibor Cinkler

Department of Telecommunications
and Artificial Intelligence
Budapest University of Technology
and Economics
cinkler@tmit.bme.hu

Katalin Friedl†

Department of Computer Science
and Information Theory
Budapest University of Technology
and Economics
friedl@cs.bme.hu

László Kabódi†

Department of Computer Science
and Information Theory
Budapest University of Technology
and Economics
kabodil@cs.bme.hu

Abstract:

The feasibility of quantum technology for solving real-world optimization problems is a question of growing interest. To investigate one side of this, we performed experiments using tools provided by D-Wave.

We consider a real life problem from telecommunications network optimization, present its ILP formulation, transform it into QUBO, and report our experimental results obtained using software libraries and (formerly) publicly accessible quantum hardware from D-Wave.

Using the default settings, we found that the freely available quantum solver did not perform as competitively as the classical tools on our test instances. Further investigation is needed to determine whether this outcome would differ for larger problems or with different settings.

Keywords: QUBO, ILP, channel assignment

1 Introduction

As quantum technology advances, practical applications to real-world problems are becoming increasingly relevant. Integer Linear Programming (ILP) is a classically hard problem, with applications in areas such as telecommunications network optimization and efficient resource allocation. ILP problems can be transformed into Quadratic Unconstrained Binary Optimization (QUBO) problems, which can be solved by quantum annealers, like those developed by D-Wave [5, 3, 4].

In QUBO, the variables are binary, and constraints have to be represented as quadratic functions and added to the objective function as penalties. When using quantum annealers to solve QUBO problems, there are some hardware-specific limitations to consider, such as limited coefficient ranges. There is also

†This paper was partially supported by the Ministry of Culture and Innovation and the National Research, Development and Innovation Office within the Quantum Information National Laboratory of Hungary.

*The research of the author was partially supported by the Doctoral Excellence Fellowship Programme (DCEP), funded by the National Research, Development and Innovation Fund of the Ministry of Culture and Innovation and the Budapest University of Technology and Economics.

a pre-processing step: the problem graph (where the variables are the vertices and the quadratic terms are the edges) must be embedded into the hardware graph (where the qubits are the vertices and the physical connections are the edges), which takes significant time.

In this paper, we describe a telecommunications network optimization problem, present an ILP formulation for it, transform it into QUBO, and give the results of our experiments solving this problem using software libraries and (formerly) openly available quantum hardware from D-Wave.

Using the default settings, we found that the freely available quantum solver does not perform as competitively as the classical tools implemented by the same company on our test instances. This might change for larger problems, which could not be embedded into the system available to us, or with better configuration, such as different annealing times, or with other strategies, such as implementing an alternative formulation of the problem. Unfortunately, D-Wave recently discontinued public free access to their hardware and our application to the new Quantum LaunchPad program was rejected, preventing us from making further experiments in these directions.

2 Channel assignment problem

Telecommunication networks are composed of cells, which contain a base station and the devices connected to it. Due to a new technology (LTE Direct), devices that are physically close enough can communicate directly, without going through the base station. Each cell uses a set of frequency channels, distinct from its neighbors to prevent cross-cell interference. Inside the cell, while the base station multiplexes regular communication, devices communicating directly on the same channel may interfere with the others. The cost of interference may be modeled with Rayleigh distribution, and with a higher cost some devices may be refused.

Assume that there are n regular users, m channels, and p directly communicating pairs. In a natural ILP formulation of the problem, binary variables $x_{i,j}$ are assigned to each $1 \leq i \leq n$ and $1 \leq j \leq m$, setting the variable to 1 if and only if the ith regular user is communicating on the jth channel. The variables $y_{k,j}$ are similar, just for the kth direct pair and the jth channel ($1 \leq k \leq p$, $1 \leq j \leq m$). Variables z_k indicate if the kth direct pair is refused ($1 \leq k \leq p$).

The objective function is

$$\min_{x,y,z} \left(\sum_{i=1}^{n} \sum_{k=1}^{p} \sum_{j=1}^{m} \alpha_{i,k,j} \cdot x_{i,j} \cdot y_{k,j} + \sum_{k \neq \ell} \sum_{j=1}^{m} \beta_{k,\ell,j} \cdot y_{k,j} \cdot y_{\ell,j} + \sum_{k=1}^{p} \gamma_k \cdot z_k \right),$$

where parameters $\alpha_{i,k,j}$ and $\beta_{k,\ell,j}$ describe the cost of interference on the jth frequency channel, between the ith regular user and kth direct pair, and between the direct pairs k and ℓ, respectively. Parameter γ_k is the cost of refusing service to the kth direct pair.

The constraints

$$\sum_{j} x_{i,j} = 1, \qquad 1 \leq i \leq n \tag{1}$$

$$z_k + \sum_{j} y_{k,j} = 1, \qquad 1 \leq k \leq p \tag{2}$$

express that there is exactly one channel per regular user (1) and either there is exactly one channel per direct pair, or the pair is turned off (2).

When transforming the problem into QUBO the constraints have to be built into the objective function using additional quadratic terms. In the case of equality constraints, it is straightforward to express them as a quadratic function which takes its minimum when the constraints are satisfied, as

$$\lambda_1 \sum_i (\sum_j x_{i,j} - 1)^2 + \lambda_2 \sum_k (z_k + \sum_j y_{k,j} - 1)^2.$$

The goal is to minimize the sum of the original objective function and this so-called penalty function. The two coefficients λ_1 and λ_2 are chosen so that constraint violations result in a large enough value of the objective function to ensure the solver avoids infeasible assignments.

3 Generating the test cases

To reduce the size of the input, in our experiments, we used a simplified version of the problem without regular devices: only the β and γ parameters, and the y and z variables were present. The function we

worked with has the form

$$\sum_{k\neq\ell}\sum_{j=1}^{m}\beta_{k,\ell,j}\cdot y_{k,j}\cdot y_{\ell,j}+\sum_{k=1}^{p}\gamma_k\cdot z_k+\lambda_2\sum_k(z_k+\sum_j y_{k,j}-1)^2.$$

To obtain reasonable settings ([1]), we created a set of test cases by randomly selecting p locations in a $500\,\mathrm{m}\times 500\,\mathrm{m}$ square region, each representing a direct pair. We assigned every pair a random transmit power between 1 W and 10 W, sampled a Rayleigh fading coefficient (scale parameter 1) to account for small-scale fading, clipped its value to the interval $[0, 1000]$, and used a path-loss exponent of 3.5, which is typical for urban environments.

Then, the interference cost $\beta_{k,\ell,j}$ between the two pairs k and ℓ on channel j is calculated as

$$\beta_{k,\ell,j}=\frac{P_k\,(f_k)^2}{(d_{k,\ell})^\alpha},$$

where P_k is the transmit power of pair k, f_k is its fading coefficient, $\alpha = 3.5$ is the path-loss exponent, and $d_{k,\ell}$ is the distance between the two pairs k and ℓ.

The γ_k cost of refusing service to pair k is set to a fixed value, two times the upper bound on the β values.

We generated one test case for each combination of p and m both ranging from 1 to 19. The generated tests have coefficients β of magnitude ranging from 10^{-5} to 10^{-9}, and $\gamma_k \approx 0.002127$ is constant for all k. The parameter λ_2 has been set to 16 in all cases, sufficiently larger than the other constants.

4 Experiments on DWave hardware

We ran experiments using the ExactSolver from the dimod library, the TabuSampler from the dwave.samplers library and the DWaveSampler from the dwave.system library. We sampled on Advantage_system4.1, with 5760 physical qubits, which has the Pegasus topology with shape 16. We used the default settings for all three solvers [2].

The plot below shows the runtimes given the variable counts of our test cases. Each test case is represented by three (or two) dots, each corresponding to the DWaveSampler, TabuSampler and, when it was small enough, the ExactSolver result.

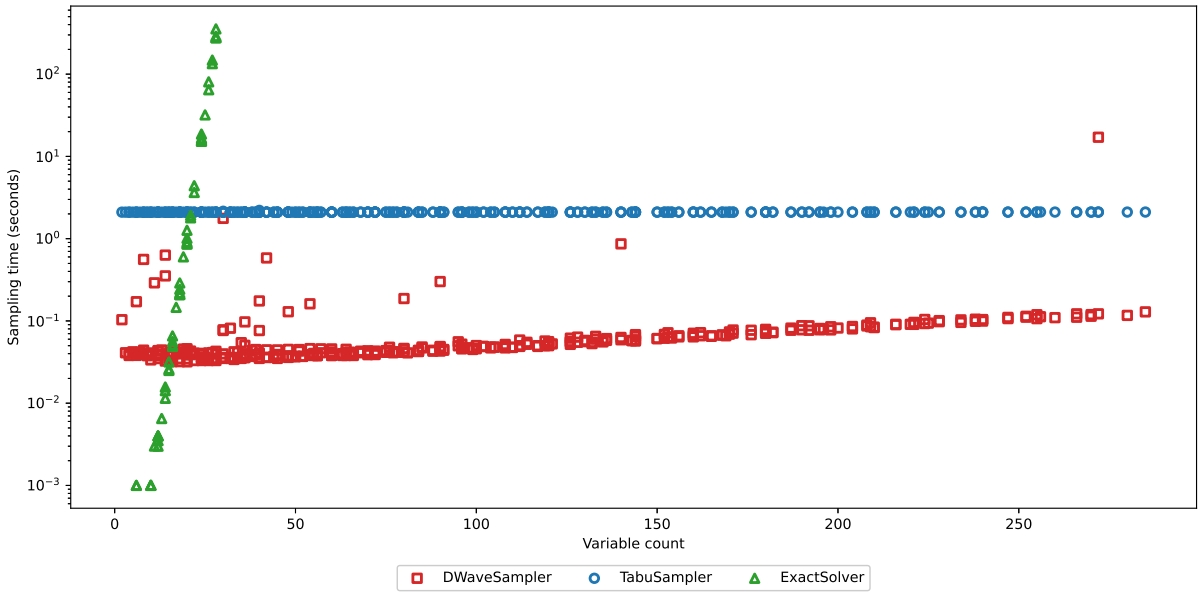

Figure 1: Solver times versus variable count

ExactSolver enumerates all possible solutions, resulting in significant runtime increases as the number of variables grows. The implementation of ExactSolver by DWave materializes large numpy arrays in memory, so we could only run the smaller test cases with it.

The TabuSampler uses the classical tabu search algorithm, that was run 100 times per test case, with a timeout parameter set to the default 20 milliseconds, resulting in around 2.1 s runtimes.

With DWaveSampler the total sampling time consists of QPU programming time, which was generally around 15.76 milliseconds in all test cases, QPU access time per sample taken (consisting of 20 μs annealing time, varying readout times, and a 20.58 μs delay time), and varying post-processing times. Readout and post-processing time increased with more variables in the test case.

In addition, running DWaveSampler requires a pre-processing step, embedding the problem graph into the hardware graph, which was done using the minorminer library from D-Wave and measured separately. Note that this has to be run only once for each test case, using a classical computer.

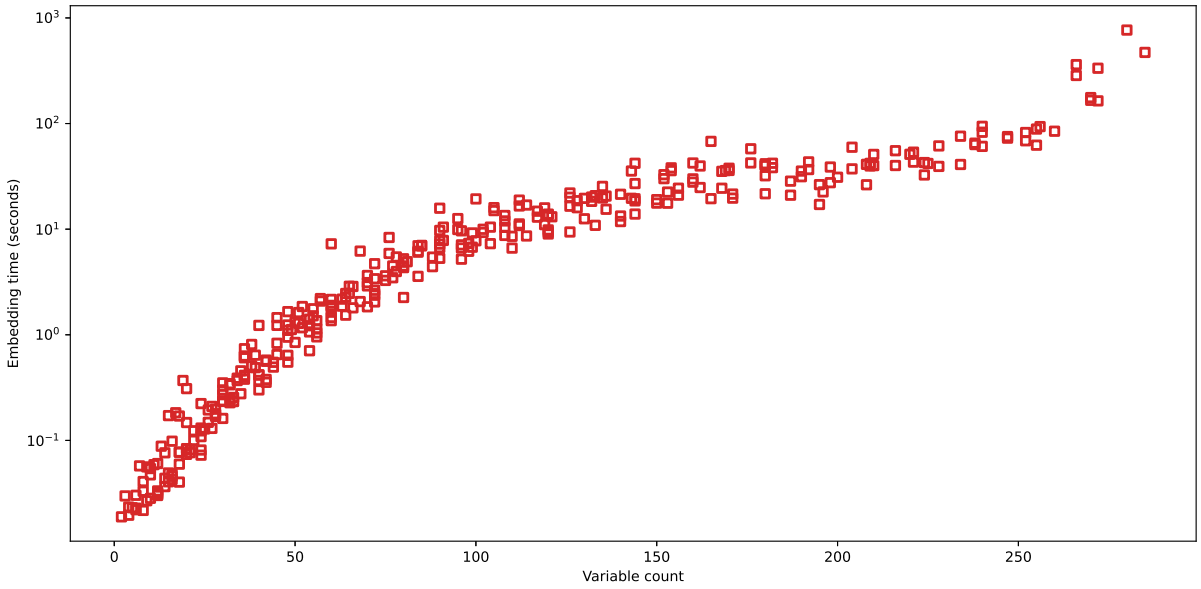

Figure 2: Embedding times versus variable count

One can see, that this is a major weakness of adiabatic solvers: the time required to perform the minor embedding of the problem graph is substantial compared to the actual time spent on the QPU, even when the same embedding is reused for several runs.

The plot below shows the solution values found, given the variable counts of our test cases.

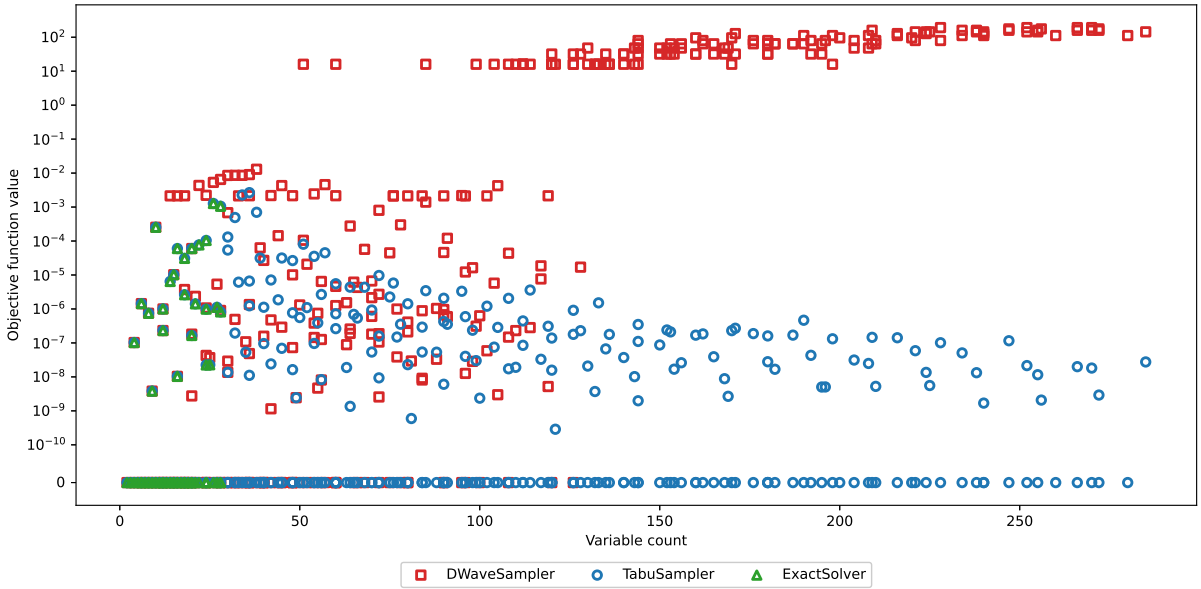

Figure 3: Solution values versus variable count

We can see a general pattern where TabuSampler seems to outperform DWaveSampler on larger test cases. We used the default timing settings for all three solvers, which resulted in TabuSampler being given roughly an order of magnitude more time to run than DWaveSampler, if we do not take the embedding time into account.

It remains an open question how the plot above would be affected by different time-limit settings. Unfortunately, we were unable to investigate this further due to the discontinuation of the free tier access to D-Wave hardware.

Now, we will look at some concrete examples and detailed comparisons.

4.1 Test cases with a single channel

First, we look at test cases where the number of channels was $m = 1$. Since the cost of refusing a pair is around $2 \cdot 10^{-3}$, while the β we generated were at most magnitude 10^{-5}, for the test cases we have, the optimal solution is to allow all to operate.

The plot below shows the solution values found for these test cases, given the variable counts.

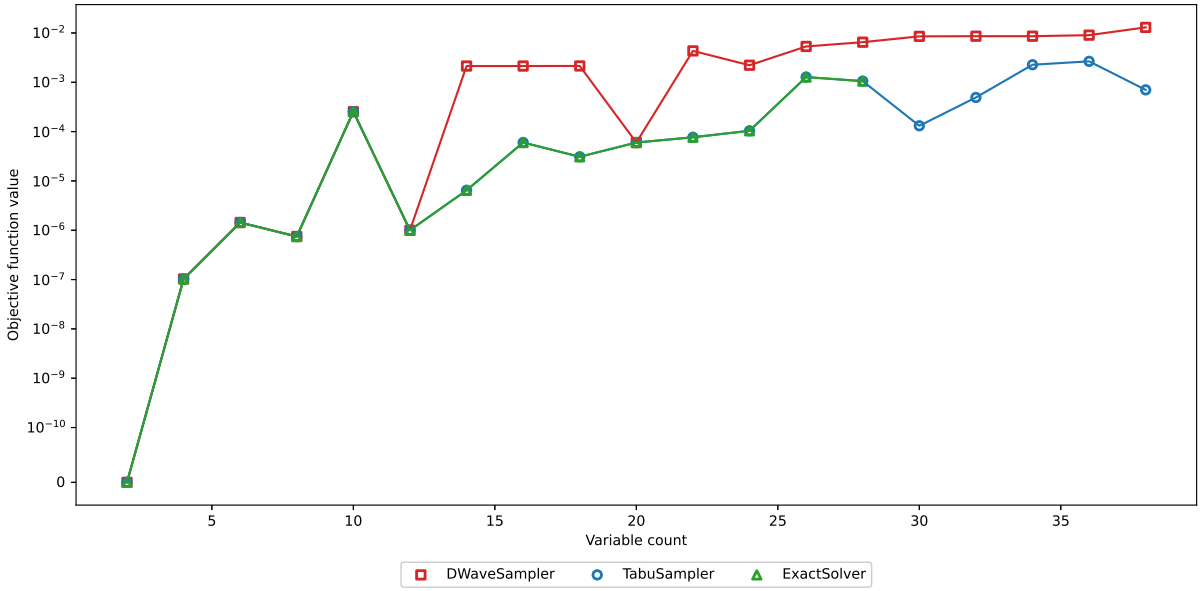

Figure 4: Solution values versus variable count for 1 frequency

TabuSampler finds the optimal solution for up to $p = 16$ pairs, then for $p = 17$ and $p = 18$ it refuses exactly one pair, which is not optimal. However, the solution found for $p = 19$ is optimal again. In contrast, DWaveSampler only finds the optimal solution up to $p = 6$. After that, DWaveSampler starts turning off some pairs unnecessarily. Here, no constraint was violated by either solver in any test case.

Since we know that DWaveSampler internally represents the quadratic function at low resolution, we theorize that its poorer performance might be due to the γ penalties being much closer in range to the β costs. As a result, for DWaveSampler it might look more advantageous to turn off some pairs. This observation has led us to experiment with a quantized version of the problem, see Section 5 for more details.

4.2 Test cases with maximum number of channels

Next, we look at cases where there are $m = 19$ channels, which means that the optimal solution is assigning all pairs to unique channels and the objective function value corresponding to that is 0.

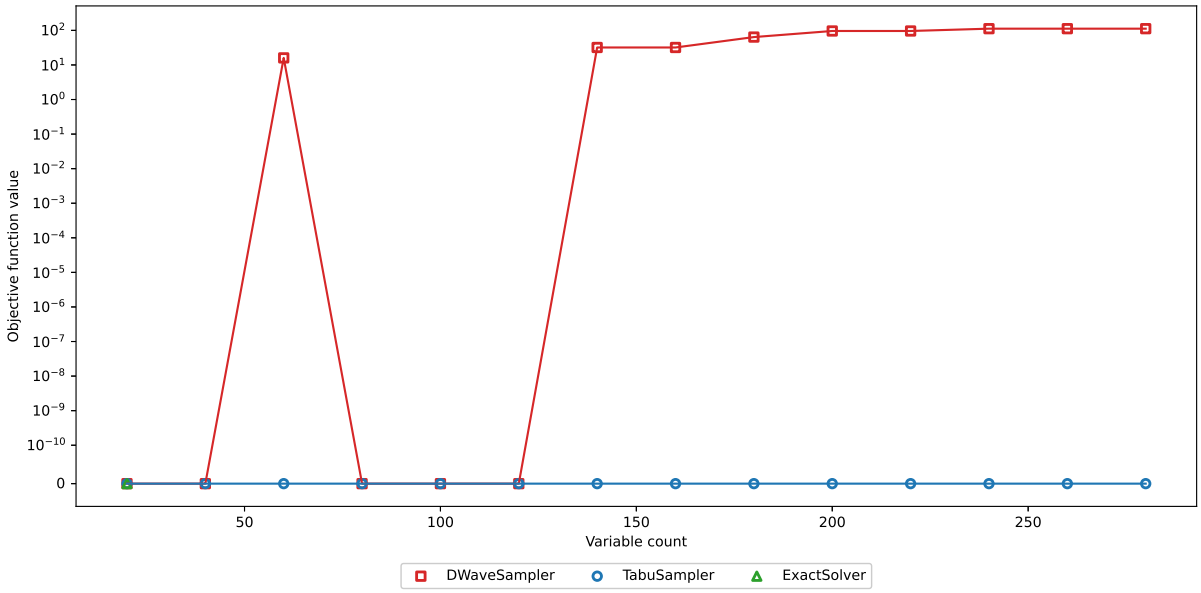

Figure 5: Solution values versus variable count for 19 frequencies

TabuSampler consistently found the optimal solution, whereas DWaveSampler failed to do so for larger instances. In cases where DWaveSampler produced a suboptimal solution, constraint violations occurred: either multiple channels were assigned to a single pair, or no channel was assigned while the pair was not refused either.

4.3 Test cases with 10 channels

Finally, let us look at a general case, where $m = 10$ channels were given.

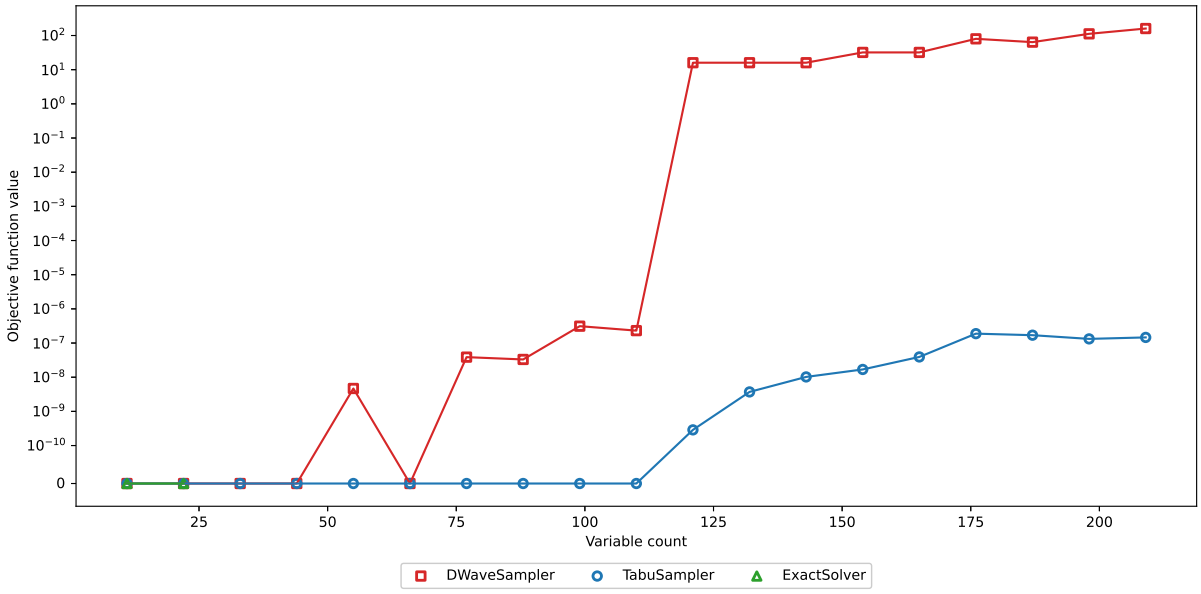

Figure 6: Solution values versus variable count for 10 frequencies

For smaller test cases, the number of channels exceeds p, the number of pairs, so the trivial solution is to assign a unique channel to each pair. As p grows larger, this is no longer possible, leading to some interference.

We observed that up to $p = 17$ TabuSampler distributed the pairs evenly across the channels. For $p = 18$, two channels were assigned 3 pairs each, while four channels were given only 1 pair, which could be better distributed. Similarly, for $p = 19$, there was one channel with 4 pairs, while three channels were

given only 1 pair. In contrast, for larger test cases DWaveSampler violated a number of constraints and the solutions typically contained unused and overused channels too.

5 Experiments with quantization

Quantum annealers have a limited coefficient range, which makes high-precision functions difficult to represent. Rayleigh fading coefficients require high precision, which might cause problems for DWaveSampler. Therefore, we also tested a quantized version of our objective function to see if restricting the coefficients to a smaller set would yield solutions from DWaveSampler that are closer to those obtained by TabuSampler. We ran both TabuSampler and DWaveSampler on the quantized objective function and compared their performance.

The objective function was quantized by mapping the β values uniformly to integers between 1 and 5, setting the γ values to 15, and keeping $\lambda_2 = 16$. This scheme was chosen because DWaveSampler can represent coefficients with up to 4-bit precision. Unfortunately, we were not able to explore alternative quantization approaches.

Quantization does not change the structure of the problem graph, so the embedding task in this experiment is identical to the previous, non-quantized experiment and could be reused.

Figures 7 and 8 show the solution values found for the quantized problem. The first one contains all data points and is comparable to the non-quantized case in Figure 3, while the second contains only non-zero values, allowing for a better resolution.

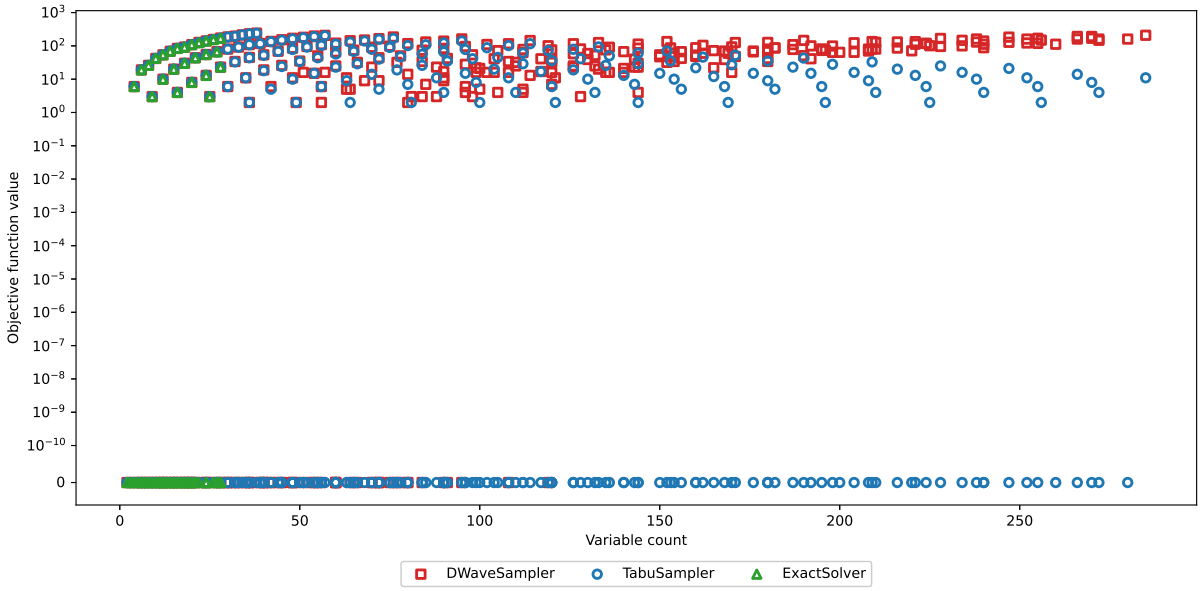

Figure 7: Solution values vs. variable count (quantized case)

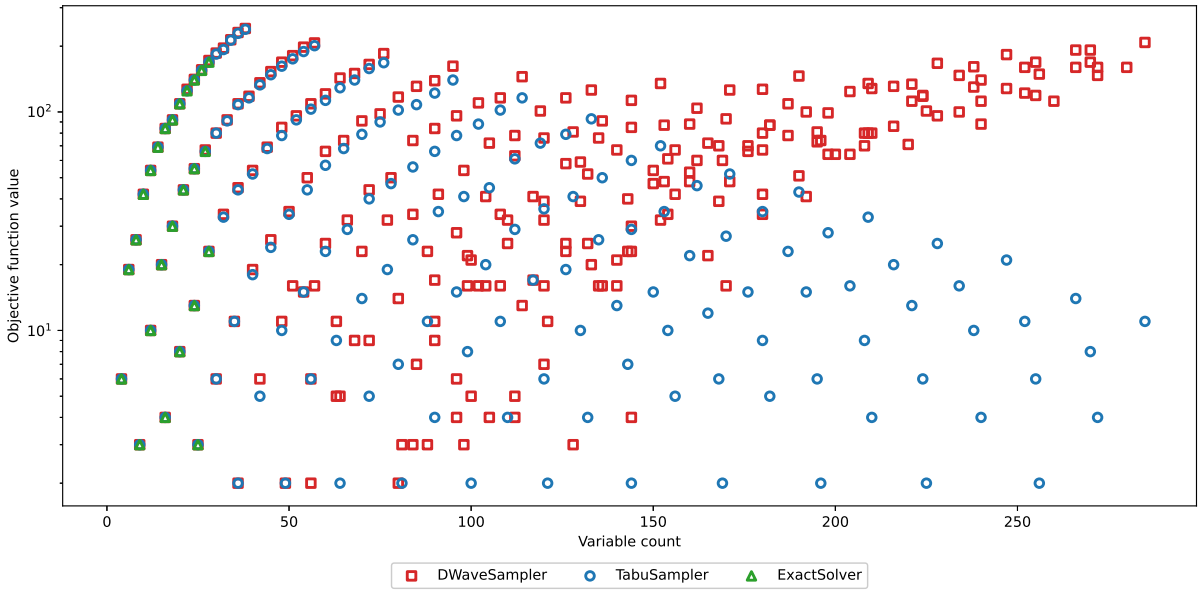

Figure 8: Solution values vs. variable count (quantized case, non-zero values only)

In general, as the number of variables grows, TabuSampler starts to find better solutions compared to DWaveSampler on these tests, even with quantized coefficients.

However, specifically for $m = 1$, TabuSampler and DWaveSampler found very similar solutions, which in this case suggests that the low coefficient range is indeed a problem DWaveSampler suffers from and is a limitation of currently available quantum annealing hardware. In these test cases, both solvers deactivated pairs and violated constraints, which now had similar penalties and were much closer to the interference costs. For $m = 10$, for the smaller cases, DWaveSampler and TabuSampler found very similar solutions. However, for the larger cases, DWaveSampler began violating constraints, and over- or underutilizing some channels. In contrast, TabuSampler violated no constraints, and began over- or underutilizing channels later. For $m = 19$ we had similar results to the non-quantized case, TabuSampler found the trivial optimal solution, while DWaveSampler violated a large number of constraints.

In summary, we believe our findings suggest that the limited resolution of current annealing hardware makes it challenging to encode constraints in the objective function. In practice, the penalty terms cannot be set sufficiently large enough to avoid infeasible solutions.

References

[1] Q. Alghazali, H. Al-Amaireh, and T. Cinkler, Graph Coloring and User Clustering-Based Resource Allocation for Device-to-Device Communication in 5G Networks, *manuscript* (2025).

[2] D-Wave Systems Inc., D-Wave libraries `https://github.com/dwavesystems` and documentation `https://docs.dwavesys.com/docs/latest/` (Accessed: February 17, 2025).

[3] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang, The unconstrained binary quadratic programming problem: a survey, *Journal of Combinatorial Optimization* **28**(1), 58–81 (2014).

[4] A. Lucas, Ising formulations of many NP problems, *Frontiers in Physics* **2** (2014).

[5] A. P. Punnen, Introduction to QUBO, in *The Quadratic Unconstrained Binary Optimization Problem: Theory, Algorithms, and Applications* (ed. A. P. Punnen), Springer (2022).