On the Design and Characterization of Set Packing Problem on Quantum Annealers

Marco Venere[1], Giuseppe Sorrentino[1], Beatrice Branchini[2], Davide Conficconi[2], Elisabetta Di Nitto[2], Donatella Sciuto[2], Marco D. Santambrogio[2]
Dipartimento di Elettronica, Informatica e Bioingegneria, Politecnico di Milano, Milan, Italy
[1]{firstname.lastname}@mail.polimi.it; [2]{firstname.lastname}@polimi.it

***Abstract*—The beginning of the quantum-era has increased the hype around computationally demanding problems and, particularly, around NP-complete problems. One of the most famous is Set Packing, which has several applications in different management contexts. Considering this, the present work provides a solver for Set Packing Problem on quantum annealers exploiting the QUBO model. In addition, It introduces a general format to express instances of the Set Packing Problem that lowers the barriers to adopting quantum annealers and enables an automatic characterization. Comparing the performance of two different quantum annealers topologies, we highlighted the pros and cons of both technologies. As key evaluation indicators, this work uses spatial features, such as the number of qubits and maximum chain length, and the execution time required for a reliable problem resolution and its breakdown.**

***Index Terms*—Quantum Annealers, Set Packing Problem, optimization problems, problem format.**

I. Introduction

Over the last decades, with the always-increasing demand for resource management and logistic constraints, cutting-edge computer science research has focused on developing operations research models and algorithms to optimize and automate decision-making processes. These models consist in maximizing a profit or minimizing a loss for a set of problem-specific parameters and constraints [1]. Among the most spread optimization tasks, the *Set Packing Problem* is an NP-complete decision problem which, given a *universe* set and a list of subsets as input, aims at finding a collection of mutually disjoint subsets, while maximizing the total number of subsets included. It finds plenty of application fields, such as railway planning [2], sphere packing [3], and airline crew scheduling [4].

Given the highly complex nature of these problems, existing tools can deal with them efficiently only at the cost of reducing the degrees of freedom, hence, obtaining suboptimal results. An example of software is CPLEX [5], which targets a wide range of programming problems. An improvement against classical solutions based on CPLEX consists of solving the problem heuristically with algorithms like Greedy Randomized Adaptative Search Procedure [6]. Even though this method is more effective than the classical solutions, issues and time constraints with complex instances of the Set Packing Problem are far from negligible. However, Set Packing Problems are only a small portion of complex optimization problems whose solutions cannot be found easily. Other examples are the Maximum Cut, the Quadratic Knapsack, and the Minimum Vertex Covering Problems [7].

In order to find an efficient solution, over the past ten years, scientific research has focused on looking for solutions exploiting a new emerging architecture, namely quantum computers. Their model of computation involves quantum principles like superposition, entanglement, and interference [8]. Thanks to these principles, it is possible to achieve a higher degree of parallelism and obtain up to superpolynomial speedup compared to classical algorithms.

Nowadays, two main quantum computation models emerge above the others: gate-based quantum computers [9] and quantum annealers [10]. However, one of these systems' main drawbacks is noise. We are in the so-called Noisy Intermediate-Scale Quantum (NISQ) era, and noise represents one of the most challenging problems in modern quantum computers. Due to noise, gate-based NISQ devices can work with only a reduced amount of qubits, limiting their applicability in real complex problems. Instead, the annealer model allows devices with a greater number of qubits within the system, without excessively suffering from noise, at the cost of a fixed architecture. However, researchers have proved that by exploiting the Ising model [11] for statistical mechanics and the Quadratic Unconstrained Binary Optimization (QUBO) [12], optimization problems can be instantiated on quantum annealers to achieve a boost in performance.

In the depictured scenario, this paper presents the design of a solver for the Set Packing Problem using quantum annealers. More specifically, we used a cloud-based platform giving application developers real-time access to D-Wave quantum computers. We exploited the well-known QUBO formulation provided by Glover et al. [7] to provide an effective solution. Using such a complete formulation allowed us to obtain a solution for the Set Packing Problem without limiting any degree of freedom. Furthermore, in favor of providing a general solution, the present paper also presents a general format to describe the Set Packing Problem.

For these reasons, our main contributions are:

1) The design of an open-source code for D-Wave quantum

DOI 10.1109/EUROCONXXXXX.2023.XXXXXXX

annealers to solve the Set Packing Problem without limiting degrees of freedom (§III-A);
2) The definition of a universal format for problem instances representation, which is currently missing in the literature. Such a format also allows to express different importance levels for each subset as weights (§III-C);
3) A computational comparison between two D-Wave quantum annealers in terms of the number of qubits, maximum chain length, chain strength, minimum energy, and time required to solve the problem (§IV).

II. Mathematical and Quantum Background

A. Set Packing Problem and its QUBO Formulation

The Set Packing Problem is a combinatorial programming problem extensively studied in recent years [13]. The formulation of the problem is the following [14]:

Definition 1 (Set Packing Problem)**.** *Given a collection $\mathcal{S}$ ($|\mathcal{S}| = n$) of finite subsets of the universe $\mathcal{U}$, a packing is a subcollection $\mathcal{S}' \subseteq \mathcal{S}$, all members of which are mutually disjoint. The Set Packing Problem is to find a packing of maximum size.*

This definition allows different model formulations, for instance, the classical or the QUBO ones. Indeed, Glover et al. [7] reformulated the Set Packing Problem under the QUBO modeling framework, describing the binary optimization problems in terms of:

$$minimize/maximize\ y = x^t Q x$$

where x is a vector of binary decision variables and Q is a square matrix (symmetric or upper triangular) of constants. In the QUBO model, the traditional constraints of the optimization problem are mapped onto a set of quadratic penalties. For example, given a minimization problem with a constraint $x_1 + x_2 \leq 1$, the objective function will become $minimize\ y = f(x) + Px_1x_2$, with P chosen sufficiently large. The main strength of the QUBO formalization is its adaptability to several optimization problems and the ease of implementation onto different hardware substrates, such as neuromorphic computers and quantum annealers [15]. For the Set Packing Problem, we obtained the following QUBO representation:

$$max\ x^t Q x$$

That corresponds to the following:

$$max \sum_{j=1}^{n} w_j x_j; \quad such \quad that \quad \sum_{j=1}^{n} a_{ij} x_j \leq 1 \forall i \in 1..m$$

where: n is the cardinality of $\mathcal{S}$, i.e., the number of possible subsets of the universe $\mathcal{U}$; m is the number of elements of $\mathcal{U}$; x_i are binary variables representing the selection of the subset i in $\mathcal{S}$ (1 if the subset is selected, 0 otherwise); w_j are weights associated with each subset in $\mathcal{S}$, representing their importance in the selection process, and thus the probability to be taken in the final solution; a_{ij} are binary coefficients which are valued one if subset j contains element i. They represent a *penalty* in the selection process to take non-disjoint-ness between subsets into account. According to this formulation, the framework is built as an optimization problem, which is also a suitable problem formulation for anneal-based quantum computers.

B. D-Wave Quantum Annealers

To execute the algorithm, we employed the quantum annealers developed by D-Wave Systems [16], which are accessible through Leap, the real-time Quantum Application Environment provided by the company. We considered two quantum annealers, the 2000Q and the Advantage, that work at extremely low temperature (around 15 mK) [17]. These devices both present a D-Wave Quantum Processing Unit (QPU), which is a lattice of interconnected qubits that uses specific interconnections network among qubits. Specifically, these are performed via internal and external couplers, which are devices that can make two qubits tend to assume the same state – both 0 or 1 – or opposite states. Between coupled qubits, it is possible to program a correlation weight that allows strengthening the link between them. This can be done by setting the coupling strength, which is one of the program parameters better described in §III [18].

The considered architectures' QPU is not fully connected, but can be connected by internal or external couplers. Specifically, 2000Q's configuration is named *Chimera*, uses 2000 qubits and 6000 couplers. Internal couplers are used to link adjacent qubits. This set of linked qubits creates a unit. In 2000Q, this linking happens following two possible representations depicted in Figure 1: column or cross rendering of qubits. These are essential to distinguish ways to organize and connect qubits. In Advantage, instead, the configuration, named *Pegasus*, uses 5000 qubits and 35000 couplers. Its internal couplers connect qubits with opposite orientations and there is not a simple schema as the one of 2000Q. External couplers are instead used to connect colinear pairs of qubits in different cells [19].

The configuration of couplers can be evaluated in terms of the degree of connectivity for each qubit, expressesing the average number of connected qubits for each of them. For the configurations considered, it is 6 for 2000Q and 15 for Advantage. Couplers topologies are generally limited, because of the reduced connectivity between qubits. This implies that, in order to connect a higher number of qubits with each other, the QPU creates *chains* among qubits. In a chain, more qubits represent the same logical variable. As an example, with a 6-degree connectivity, a connection between one variable and 7 different variables requires a chain, in which two qubits, linked to each other, will represent the same logic element; the first one will be then connected to 3 other qubits, and the second one to the 4 remaining qubits.

III. Proposed Approach

This Section describes the approach for the resolution of Set Packing Problem. Figure 2 depicts the proposed workflow schema that aims to cover the definition of specific problem instances and their resolution. In particular, the user provides details about the instance of the Set Packing Problem and

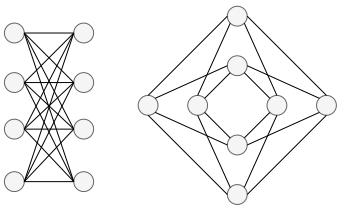

Fig. 1: Chimera topology: column rendering (left) and cross rendering (right) of qubits (vertices) and couplers (edges)

the automatic problem generator creates the JSON version of that will be read by the solver. Another possibility of the automatic problem generator is to randomly create new instances of the SetPacking problem given a certain number of variables and constraints. This allows to test the framework in more general situations. §III-A describes the solver algorithm and its implementation exploited to resolve a specific problem instance. Afterwards, §III-B illustrates our steps for hyperparameter tuning, that improve the overall results. Finally, §III-C details the novel format that eases the QUBO representation generation and the automation of D-Wave quantum annealers characterization, while lowering the barriers required to use such novel technology.

A. The Solver Algorithm

As specified by Glover et al. [7], adopting the QUBO formalization enables the exploitation of quantum annealers. Therefore, we adopt this approach to design the solver algorithm and its implementation. Since we are using D-Wave quantum annealers, we took advantage of specific D-Wave libraries, which allowed us to implement and solve a QUBO problem. Among the several programming tools provided by D-Wave, Ocean [20] is a software suite containing tools for resolving hard mathematical problems. Specifically, Ocean's Binary Quadratic Model (BQM) API [21] represents the most proper instrument to use for QUBO implementations.

Our starting point is the construction of a BQM object which defines the QUBO *variables* to be instantiated on the qubits of the used architecture and the *interactions* between them. Indeed, the variables will correspond to all the possible subsets of the problem, while the interactions will represent the constraints among subsets. As described in §II-A, each variable has its own weight, which will be the one considered to select or discard the variable. In particular, QUBO variables have negative weights associated to specify the importance of the various subsets in the solution. Instead, constraints represent the requirement that no overlapping subsets are present in the same solution. Therefore, we must define an interaction among all subsets with at least one common element, by using a positive weight. Notice that it is positive since it represents a non-acceptable combination of subsets.

Algorithm 1 describes the proposed approach. Once the BQM object is instantiated, the sampler is invoked. This object of the library performs the instantiation of the given QUBO problem on the QPU and performs the sampling process,

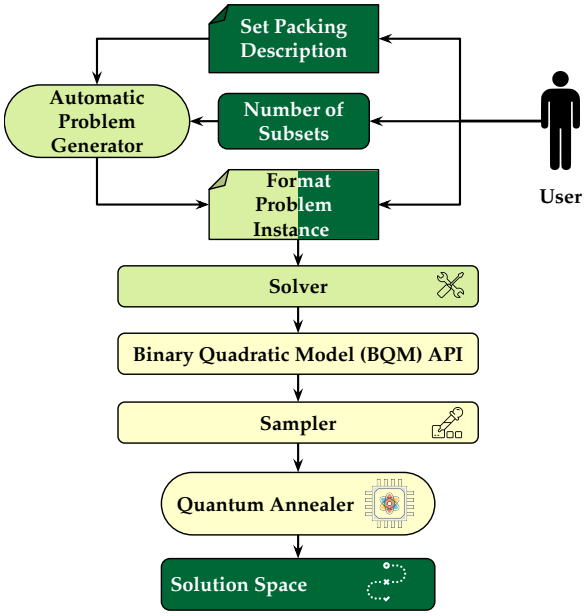

Fig. 2: High-level view of the proposed approach: light green are our contributions, while inputs/outputs are darker green.

Algorithm 1 Set Packing Problem solver

```
1:  procedure SolveSetPackingProblem(filename, N)
2:      parameters
3:          penalty: penalty for violation of constraints
4:          sampler: the sampler to use
5:      end parameters
6:      global variables
7:          P: SetPackingProblem
8:          BQM: BinaryQuadraticModel
9:      end global variables
10:     P := read_sanitized_file(filename)
11:     for each subset ∈ P.subsets do
12:         BQM.add_variable(subset.name, subset.weight)
13:     for each constraint ∈ P.constraints do
14:         for each i ∈ constraint do
15:             for each j ∈ constraint, j < i do
16:                 BQM.add_interaction(i, j, penalty)
17:     sampleset := sampler.sample(BQM,num_reads)
18:     return sampleset
```

obtaining a solution for the instance of the problem. Such a process is the core principle of a quantum annealer: the qubits and their weights on the QPU are associated with the variables of the problem. Couplers connecting qubits, instead, keep the penalties for the undesired interactions. Then, QPU applies quantum principles, like quantum fluctuations and quantum tunneling [22], to obtain a final configuration of the system, in which the potential energy function of the defined quantum topology is minimized. The minimization of such energy considers the weights of qubits and the interactions among qubits. Negative weights will reduce the overall energy,

while positive weights will increase it. Thanks to the negative weights of the qubits, the minimal solution will contain the highest possible number of qubits, i.e., subsets. Differently, the positive weights of the couplers for overlapping subsets exclude configurations violating the minimal solution. Indeed, their positive penalty would increase the overall energy of the configuration. Finally, the solution to the Set Packing Problem is the minimum point for the objective function.

At the end of the process, the sampler returns a sample set. Its first element is the configuration with minimum energy and the problem solution, while the other values represent the remaining executions. In fact, because of the probabilistic nature of the quantum annealer, more executions of the problems must be done to obtain reliable results. This number of executions, namely **num_reads**, can be specified programmatically and is one of the sampler object parameters. Instead, the output of the sampler is a set of variables that can be 1 or 0, whether the variable is taken or not.

B. Hyperparameter Tuning

Based on the designed algorithm, we had to tune some hyperparameters to improve performance and to face the quantum annealing programming issues. Among the most important hyperparameters, there is the **chain_strength**. As reported in §II-B the number of connections per qubit is related to its degree of connectivity. When more links are required, the QPU creates chains. Yet, a chain may *break* if, after the sampling process, the qubits belonging to that chain do not end up in the same state. If this event happens, the final configuration on the QPU will generally not correspond to a valid solution to the problem, because of the constraint violation. For these reasons, the **chain_strength** value requires the QPU to spend more energy and avoid chain breaks. However, we must stress the fact that choosing an excessively low or high value of chain strength has negative effects, such as a high number of chain breaks or a change of the problem nature [23]. This latter event may occur since QUBO constraints weights might be modified and combined with excessive chain strength would shrink the weights close to zero. Though this ensures no chain breaks, it solves a completely different problem.

To choose a proper chain strength value, we used the **uniform_torque_compensation** method [24], which computes the chain strength that attempts to compensate for the torque that would cause the chain breaks. Such a function receives as input the **pre_factor** parameter, which is the parameter that effectively needs to be tuned for an optimal chain strength. The default value is 1.414, but we experimentally derive 2.0 as the optimal value for our implementation.

We also tune the **num_reads** that defines the number of samples retrieved by the sampler. Though its default value is set to a single sample, we deemed that 100 samples were necessary to ascertain the correctness of the solution, given probabilistic quantum annealer outputs.

C. Proposed Format

To characterize the proposed solution scaling with the problem size, we designed a novel formal descriptive paradigm that eases the QUBO representation generation and the automation D-Wave quantum annealers chracterization. This format describes instances of Set Packing Problems, enabling the rigorous expression of all the subsets belonging to the set and the related constraints according to the Glover formalization [7]. Additionally, our novel format completely hides the complexity of programming quantum annealers. Thus, such an abstraction layer enables inexpert users to easily define and solve Set Packing problems on D-Wave quantum annealers, thus lowering the barriers to this technology.

For implementation purposes, we adopt the convenient JavaScript Object Notation (JSON) [25] format, as it is strongly expressive, widely supported by the majority of scripting languages, and efficiently parseable by compiler tools [26]. More in detail, a generic document for Set Packing will define an array of objects, each of which describes an instance of a problem. The usage of arrays allows for multiple instances to be considered and solved in the same execution of the algorithm. Each of these objects will contain two fields: *subsets*, which is an array of all the subsets considered for the universe set $\mathcal{U}$ (i.e., the collection $\mathcal{S}$) and *constraints*, which is an array of the constraints among the subsets. Such a description, therefore, does not require the items of the universe, since the variables of the QUBO formalization are the subsets. These latter are defined as arrays of JSON objects, each containing a *name* field, identifying the specific subset, and, optionally, the corresponding *weight* (set by default to 1). The *constraints* are arrays of JSON objects as well, each defining a field, *sets*, that represents a list of non-disjoint, therefore incompatible, subsets referred to by their identifiers. The criteria to construct the identifiers of the subsets is arbitrary (e.g., the decimal conversion of the one-hot encoding of the list of considered items, if m is sufficiently small).

As final remark, this format can potentially be generalized to more optimization QUBO-suitable problems, since it enables easy variables and constraints requirements.

IV. Experimental Setup and Results

We characterize the performance of Advantage and 2000Q D-Wave quantum annealers in terms of resource usage (§IV-A) and execution times (§IV-B). We tested runs with randomly generated instances of Set Packing Problem thanks to our novel format. Each run is performed 10 times to collect more reliable data. These runs have a number of constraints equal to $n/2$, where n is the number of variables. This dependency avoids random degenerative cases with too few constraints.

A. Spatial Results: Resource and Energy Scaling

Firstly, we consider the **number of qubits** and the **maximum chain length**. Since the number of qubits limits the feasibility on available quantum architectures, Table I showcases our scaling analysis against the instances sizes growth. Table I puts again a spotlight on the main difference between

TABLE I: Spatial comparison between 2000Q and Advantage

	# Qubits		Max Chain Length	
# Variables	2000Q	Advantage	2000Q	Advantage
10	25	15	3	2
20	115	55	7	4
30	269	109	12	5
40	534	214	18	7
50	794	316	22	9
60	1151	427	28	12
70	-	553	-	11
80	-	761	-	13
90	-	920	-	13
100	-	1324	-	19
110	-	1785	-	25
120	-	1804	-	22
130	-	2356	-	28
140	-	2327	-	25
150	-	3309	-	34

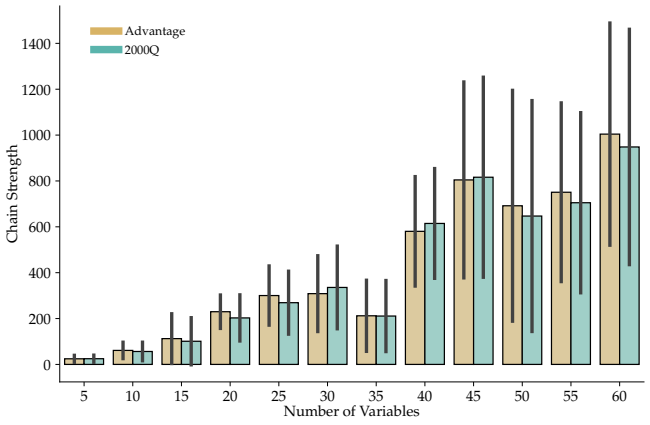

Fig. 3: Chain Strength scaling of up to 60 variables

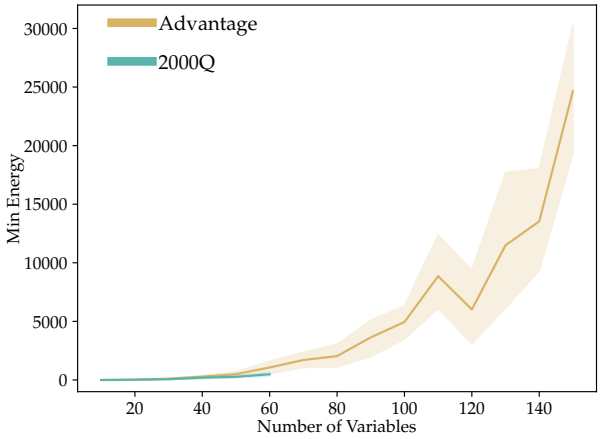

Fig. 4: Minimum Energy scaling of the considered annealers

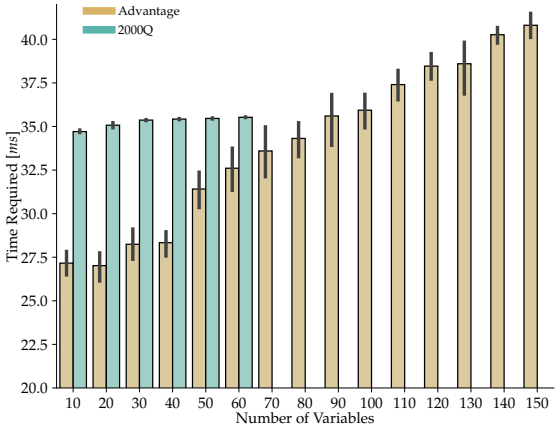

Fig. 5: QPU Access Time comparison between the annealers

Advantage and 2000Q: the number of available qubits. The former in fact has over 5000 qubits while the latter almost 2000. This leads Advantage to solve more demanding problems in terms of qubits. In fact, it can scale up to 150 variables while 2000Q can't go over 60 variables. Generally, Advantage requires fewer qubits than 2000Q for the same task, due to the better connectivity degree and couplers of *Pegasus* configuration. Instead, the chain length provides insights into solution robustness, as longer chains break more easily. Table I demonstrates that Advantage has shorter chains. This corresponds to a higher connectivity degree for its qubits. Theoretically, Advantage improved coupling strategy enables the resolution of more complex problems with smaller chains.

We also expect Advantage's better coupling to have smaller chain strength for chain break avoidance. Instead, Figure 3 graphically demonstrates experimentally that the two quantum annealers scale similarly for those problem sizes. The reason behind this stands in the higher noise of Advantage. This noise leads to higher-than-expected chain strength value from **uniform_torque_compensation**, thus not exploiting the $Pegasus$ couplers configuration (§II-B). Considering the standard deviation of the chain strength, this factor depends on the problem complexity, hence scaling agnostically to the target annealer. Besides, Figure 4 illustrates the minimum energy needed for the resolution. Indeed, the higher required chain strength of Advantage mirrors also in the required energy and its variance that is higher than 2000Q.

B. Quantum Annealing Execution Time Analysis

Execution time for the Set Packing Problem resolution on our target quantum annealers is affected by many contributions, referred to, on the whole, as **QPU Access Time**. Among the most prominent contributions, there are the **QPU Programming time** and the **QPU Sampling Time**. The former is related to the time required to prepare the BQM object and to program the QPU, while the latter is the time to solve all the problem samples. Table II reports the numerical data in μs and the D-Wave tool (called Inspector) offers a resolution of 0.01 μs [27]. The table showcases that, for problem sizes up to the 2000Q limit, the programming and sampling times of the annealers scale similarly, with a higher standard deviation for Advantage. Figure 5 illustrates

TABLE II: Execution time breakdown and scaling against the problem size of 2000Q and Advantage quantum annealers.

# Variables	QPU Access Time [μs]		QPU Programming Time [μs]		QPU Sampling Time [μs]	
	2000Q	Advantage	2000Q	Advantage	2000Q	Advantage
10	34708.60 ± 135.750	27160.44 ± 1167.398	10772.80 ± 17.214	15067.64 ± 1.262	23935.8 ± 132.183	12092.8 ± 1167.299
20	35074.86 ± 247.232	27017.68 ± 1340.257	10884.26 ± 17.834	15067.68 ± 1.237	24190.6 ± 257.968	11950.0 ± 1339.747
30	35367.48 ± 43.545	28242.32 ± 1482.749	10965.68 ± 37.49	15067.92 ± 1.416	24401.8 ± 67.714	13174.4 ± 1483.259
40	35425.20 ± 50.101	28332.92 ± 1118.501	11004.00 ± 52.44	15068.52 ± 1.688	24421.2 ± 37.312	13264.4 ± 1118.157
50	35462.34 ± 47.403	31412.84 ± 1748.458	11019.94 ± 38.682	15069.44 ± 1.449	24442.4 ± 11.384	16343.4 ± 1747.558
60	35527.96 ± 24.363	32606.36 ± 2127.879	11082.36 ± 24.422	15070.16 ± 2.517	24445.6 ± 0.843	17536.2 ± 2126.586
70	-	33596.56 ± 2472.715	-	15069.96 ± 2.3851	-	18526.6 ± 2472.826
80	-	34314.52 ± 1692.575	-	15070.12 ± 2.510	-	19244.4 ± 1693.747
90	-	35603.28 ± 2528.003	-	15072.48 ± 3.087	-	20530.8 ± 2527.003
100	-	35934.20 ± 1625.577	-	15070.00 ± 3.157	-	20864.2 ± 1626.582
110	-	37404.88 ± 1481.973	-	15071.68 ± 4.626	-	22333.2 ±1482.099
120	-	38464.40 ± 1320.063	-	150703.20 ± 3.112	-	23391.2 ± 1319.568
130	-	38598.84 ± 2651.135	-	15070.84 ± 3.817	-	23528.0 ± 2649.242
140	-	40265.40 ± 809.774	-	15073.60 ± 2.128	-	25191.8 ± 809.278
150	-	40807.88 ± 1198.013	-	15075.28 ± 3.991	-	25732.6 ± 1196.120

that Advantage is significantly faster than 2000Q for all the tasks solved, while its variance, has a noteworthy increase in contrast with 2000Q stability. This phenomenon is imputable to the noise that affects Advantage.

To summarize the experimental characterization, Advantage enables the resolution of more complex problems than 2000Q, while scaling chain strength, energy, and time with many noise-related issues. Indeed, the variance, measured again as standard deviation, in solving big problem sizes is far from negligible. Conversely, 2000Q tackle smaller problem instance with a slower yet stable behavior. For these reasons, 2000Q would be the way to for smaller problem sizes, while bigger ones can exploit Advantage architecture.

V. Conclusion And Future Works

This manuscript presented a general formulation and a solution methodology for the Set Packing Problem on D-Wave quantum annealers. We designed a novel format that eases the QUBO generation and the automation of annealers characterization while lowering the barriers to using quantum annealer technology. We analyze the strengths and weaknesses of Advantage and 2000Q annealers. 2000Q exhibits higher execution times with more stable results. Instead, Advantage has a higher computational power and can scale the problem size, but presents a non-negligible noise. We analyzed how to tune hyperparameters to improve quantum annealers exploitation.

Based on this work, we envision new research directions. On the one hand, Glover's formulation, our solver, and format pave the way for a library to solve optimization problems through quantum annealers. In this regard, this library could be a novel abstraction layer that completely hides the complexity of using quantum annealers while offering an easier entry point. On the other hand, we envision evaluating our characterization framework on different quantum annealers. Finally, we will also work towards a comparison with quantum computers based on the gate paradigm while finding a methodology to map the annealing problem on such architectures.

Acknowledgments

This work has financial support from ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union – NextGenerationEU.

References

[1] W. Winston and J. Goldberg, *Operations research: applications and algorithms*, 01 2004.

[2] Y. Gao, L. Yang, and S. Li, "Uncertain models on railway transportation planning problem," *Applied Mathematical Modelling*, vol. 40, no. 7, pp. 4921–4934, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0307904X15008252

[3] T. C. Hales, "The sphere packing problem," *Journal of Computational and Applied Mathematics*, vol. 44, no. 1, pp. 41–76, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/037704279290052Y

[4] J. P. ARABEYRE, J. FEARNLEY, F. C. STEIGER, and W. TEATHER, "The airline crew scheduling problem: A survey," *Transportation Science*, vol. 3, no. 2, pp. 140–163, 1969. [Online]. Available: http://www.jstor.org/stable/25767518

[5] C. Optimization, "Cplex callable library-base system with barrier and mixed integer solver options," 1995.

[6] T. A. Feo and M. G. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations research letters*, vol. 8, no. 2, pp. 67–71, 1989.

[7] Y. D. Fred Glover, Gary Kochenberger, "Quantum bridge analytics i: A tutorial on formulating and using qubo models," *4OR*, 2019, May.

[8] M. P. Lobo, "Quantum superposition as entanglement," 2019.

[9] Q. Flagship, "Gate based quantum computers," https://qt.eu/discover-quantum/underlying-principles/gate-based-qc/.

[10] D–Wave Systems Inc, "D-wave-hybrid framework documentation," https://docs.dwavesys.com/docs/latest/c_gs_2.html.

[11] A. Galluccio, M. Loebl, and J. Vondrák, "New algorithm for the ising problem: Partition function for finite lattice graphs," *Physical Review Letters*, vol. 84, no. 26, p. 5924, 2000.

[12] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using qubo models," 2018. [Online]. Available: https://arxiv.org/abs/1811.11538

[13] G. Gottlob and G. Greco, "Decomposing combinatorial auctions and set packing problems," *journal of the ACM*, vol. 60, no. 4, August 2013.

[14] V. T. Paschos, "A survey of approximately optimal solutions to some covering and packing problems," *ACM computer survey*, vol. 29, no. 2, June 1997.

[15] J. B. Aimone, K. E. Hamilton, S. Mniszewski, L. Reeder, C. D. Schuman, and W. M. Severa, "Non-neural network applications for spiking neuromorphic hardware." Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2018.

[16] D–Wave Systems Inc, "The d-wave 2000q™ quantum computer technology overview," https://dwavejapan.com/app/uploads/2019/10/D-Wave-2000Q-Tech-Collateral_1029F.pdf, 2019.
[17] ——, "How cold is it?" https://support.dwavesys.com/hc/en-us/articles/360003681034-How-Cold-Is-It-.
[18] ——, "What is quantum annealing?" https://docs.dwavesys.com/docs/latest/c_gs_2.html.
[19] ——, "Chimera topology," https://docs.dwavesys.com/docs/latest/c_gs_4.html#topology-intro-chimera.
[20] ——, "Samplers," https://docs.ocean.dwavesys.com/projects/system/en/stable/reference/samplers.html.
[21] ——, "Binary quadratic models," https://docs.ocean.dwavesys.com/en/stable/concepts/bqm.html.
[22] ——, "What is quantum annealing?" https://docs.dwavesys.com/docs/latest/c_gs_2.html.
[23] ——, "Programming the d-wave qpu: Setting the chain strength," https://www.dwavesys.com/media/vsufwv1d/14-1041a-a_setting_the_chain_strength.pdf.
[24] ——, "D-wave system documentation," https://docs.ocean.dwavesys.com/projects/system/en/stable/reference/generated/dwave.embedding.chain_strength.uniform_torque_compensation.html.
[25] JSON, "Json," https://www.json.org/json-en.html.
[26] D. Lemire, "Parsing json really quickly: Lessons learned," https://www.infoq.com/presentations/simdjson-parser/.
[27] D–Wave Systems Inc, "D-wave inspector," https://docs.ocean.dwavesys.com/projects/inspector/en/latest/.