

QC-Adviser: Quantum Hardware Recommendations for Solving Industrial Optimization Problems

Djamel Laps-Bouraba ¹, Markus Zajac ², and Uta Störl ¹

Abstract: The availability of quantum hardware via the cloud offers opportunities for new approaches to computing optimization problems in an industrial environment. However, selecting the right quantum hardware is difficult for non-experts due to its technical characteristics. In this paper, we present the QC-Adviser prototype, which supports users in selecting suitable quantum annealer hardware without requiring quantum computing knowledge.

Keywords: Decision making, Resource estimation, Quantum annealing, Optimization problems



1 Introduction


Quantum computers have become increasingly important in recent years. Cloud providers that make quantum hardware available to everyone are increasingly attracting the interest of the industry. Companies and consortia are identifying potential applications for quantum computing in various sectors, such as logistics or material science [Ba21; Si24].

Today's quantum computers are in their infancy. They are limited in the number of qubits and produce errors in their calculations [HC23; Si24]. Therefore, ensuring a possible advantage over classical solutions is a challenge [HC23]. This makes it difficult for companies to develop business models for investments in quantum technologies [HC23], and to demonstrate the necessary business impact of quantum technologies [Ba21]. In [ZRS24], possible approaches to identifying a potential advantage when using gate-based quantum computers are discussed.

However, companies and consortia expect quantum computing technologies could have a significant business impact. Especially because research is being carried out on improved hardware [Si24]. The two expected effects are of particular interest to the industry:

1. Solution quality and efficiency: Today's algorithms often calculate a local optimum, and therefore not an optimal solution [TTK25]. Quantum technologies, such as quantum annealing, promise better solution quality and solutions for problems with large parameter ranges [Ba21]. The production and logistics environment is

¹ University of Hagen, Databases and Information Systems, Universitätsstr. 1, 58097 Hagen, Germany, info@dbouraba.de,  <https://orcid.org/0009-0006-9015-3976>;
uta.stoerl@fernuni-hagen.de,  <https://orcid.org/0000-0003-2771-142X>

² German Aerospace Center (DLR), Institute of Software Technology, 51147 Cologne, Germany, markus.zajac@dlr.de,  <https://orcid.org/0000-0002-9338-9259>

particularly affected by this, as better quality solutions contribute to process and cost efficiency.

2. Faster solutions to optimization problems: Quantum computers can achieve a speed advantage over classical solutions when solving certain problems [Aw23; Si24]. In the case of quantum annealing, the aspect of acceleration has not been clarified conclusively and is being discussed [Ya22]. The further development of improved quantum hardware could lead to enhanced devices in the future.

Regardless of the question of an advantage, the question of how to select the best possible quantum hardware currently available must also be answered. In this paper, we focus on the selection of suitable quantum annealer hardware. This type of quantum hardware is often used to solve industrial optimization problems, as Yarkoni et al. [Ya22] argue. The hardware itself is conveyed by so-called *solvers*, which are hardware resources for problem solving. For business users who are not familiar with quantum technology, it is not easy to identify a solver. They have to deal with the following questions (we briefly discuss the technical terms in Section 2):

1. How many qubits are required?
2. Can a hybrid or a QPU solver be used, or both?
3. Is a decision accompanied by a benchmark score?
4. How expensive is the use of a solver?

To support users in the decision-making process, we have developed a software prototype that we call the QC-Adviser. This dialog-based tool requests problem-specific information, makes an estimate of the resources required, and presents the most appropriate solvers to the user. The contribution of our paper is the following:

1. We describe our prototype for decision-making. This enables business users to identify suitable solvers for a range of problems.
2. We show that users hardly need any quantum computing knowledge to operate the QC-Adviser. The goal of the QC-Adviser is to hide as much specifics as possible by simple user guidance.

The remainder of this paper is organized as follows: In Section 2, we briefly cover the fundamentals. Related work is discussed in Section 3. The general workflow of the QC-Adviser prototype is introduced in Section 4. In Section 5, we demonstrate its functioning using the TSP (Travelling Salesman Problem) and provide a brief summary in Section 6.

2 Fundamentals

Quantum computers can be divided into gate-based quantum computers (circuit model) and quantum annealers (QA). QA rely on continuous-time evolution of a quantum system. Although gate-based quantum computers can handle a wider range of problems, QA are specifically designed to solve combinatorial optimization problems [Se24; Ya22]. While

classical computers are based on CPUs and bits, quantum computers are based on QPUs, which in turn contain a specific number of qubits. QA QPUs follow specific graph topologies, which in turn determine the layout and connectivity of the qubits [Ya22]. Different topologies can lead to differences in the accuracy of the results for the same problem.

In order to calculate optimization problems on a QA, they must be formulated accordingly beforehand. There is already a proven approach to this: Optimization problems can be reduced to the so-called Quadratic Unconstrained Binary Optimization (QUBO) problems [Gl22; Ya22]. It is an optimization formula that expresses a sum. The sum is based on binary variables and the products of pairs of these binary variables. Products are restricted to quadratic relationships between binary variables. In addition, each term has a coefficient.

We now turn to the TSP and its QUBO formulation as a representative example. The TSP can be described as follows: Given a set of nodes and edges (distances), the problem is to find the shortest possible tour T that visits each node exactly once and returns to the starting node [SPA22]. A tour is described as:

$$T = (p_1, \dots, p_n, p_{n+1}), p_{n+1} = p_1, \quad (1)$$

where n is the number of nodes and p_i is the node in the i th position of the tour. A binary variable is defined in the context of TSP as follows [SPA22]:

$$x_{v,p} = \begin{cases} 1 & \text{node } v \text{ is at position } p \text{ in the tour,} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The final QUBO formulation can be summarized as follows (detailed formulation can be found in the work of Stogiannos et al. [SPA22]):

$$H_{TSP} = c_1(H_{v,p} + H_{p,v} + H_{EC}) + c_2H_W \quad (3)$$

The sum terms encode the following constraints and the minimization objective:

- $H_{v,p}$: Each node must appear at exactly one position on the tour.
- $H_{p,v}$: Each position of the tour must be occupied by exactly one node.
- H_{EC} : The tour must consist of edges that really exist.
- H_W : Calculates the cost of a tour (minimization objective necessary to converge to the tour with the minimal cost).

A problem formulated as a QUBO is then embedded in a QA QPU to be solved. For the overall process, from the definition of QUBO to the readout of the result, we refer to the work of Yarkoni et al. [Ya22]. The number of qubits required for embedding can be estimated. Stogiannos et al. [SPA22] present the estimation approach in the case of the TSP.

Based on an estimate, suitable solvers can be inferred. Thereby, QPU solvers represent different QPUs with a specific number of qubits and topologies. Hybrid solvers combine

classical algorithms and QPUs. The most widely used solvers today come from the vendor *D-Wave* [TTK25; Ya22], which has several different solvers in its portfolio³.

3 Related Work

There are several works on the question of how to best select the currently available quantum hardware for a given problem.

Salm et al. [Sa23] deal with the question of how compilers and quantum computers can be automatically selected before the input circuit is compiled. To answer this question, the authors utilize machine learning methods to predict the accuracy of execution results on different quantum computers. The overall solution defines a process and an implemented software architecture based on the previous work of the authors. The *NISQ Analyzer* is the central component of the architecture [Sa20; Sa23]. This component analyzes and selects a suitable implementation and quantum computer for a quantum algorithm chosen by the user [Sa20]. This work considers gate-based quantum computers and their inputs in the form of circuits. Quantum annealers and problems formulated in QUBO have not yet been taken into account. In our work, we offer support for this case.

Poggel et al. [Po23] examine the selection of the most suitable options (such as encoding, algorithm, and quantum hardware) that must be determined for solving optimization problems with quantum computers. To this end, they developed a framework that suggests various options as solution paths. Solution paths begin with the problem formulation and end with compilation and hardware selection. The actual selection of quantum hardware can be done by a component such as the *MQT Predictor* [Po23; QBW25]. The *MQT Predictor* uses a trained model (whose training data consists of quantum circuits from a wide range of applications) to select the hardware [QBW25]. The *MQT Predictor* offers support for gate-based quantum computers. Although QUBO problem encoding is possible within the framework mentioned above, gate-based quantum hardware is selected. For this purpose, the problem is reformulated or converted as QAOA. Quantum annealers (such as those from *D-Wave*) directly supports QUBO formulations. We take advantage of this fact in our work.

In general, benchmarks can also play a role in the selection of quantum hardware [We20]. Several benchmarks have already been developed for quantum computing [Lo25]. Looking at existing benchmarks for applications, it is striking that most were originally developed for gate-based systems and for specific problems. Some, such as *Q-score*, have been extended for quantum annealing [Lo25]. *Q-score* is also used to determine the quality of the solution. We take the idea of solution quality and utilize it when selecting suitable solvers. The prerequisite is that this is known for a problem instance.

³ https://docs.dwavequantum.com/en/latest/industrial_optimization/index_get_started.html#opt-index-get-started

4 Workflow Concept

As mentioned in Section 2, the number of qubits required can be estimated. Thereupon, possible solvers can be suggested to a user on a problem-specific basis. This fact forms the basis for the general workflow of the QC-Adviser prototype, which is shown in Figure 1. Detailed description of the individual steps follows.

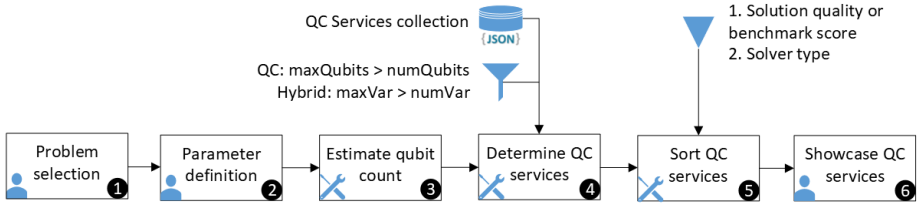


Fig. 1: The workflow contains six steps. A person symbol stands for the user's interaction with the QC-Adviser, a tool symbol stands for the calculations performed by the QC-Adviser.

Step 1. The user selects the problem to be solved and can read the description of the problem on request. Different problems are grouped into classes. The QC-Adviser currently offers the three problem classes *Routing Problems*, *Sequencing Problems* and *General Problems*. General problems are the basic problems themselves, without any extension in the form of constraints, which only contain the implicit constraints.

Step 2. Next, the user specifies the problem instance. Additional constraints can also be specified. The latter is interesting if the constraints defined by the problem definition should be modified. Problem-specific forms support the user in specifying the constraints.

Step 3. From the information in the previous step, this step estimates the number of qubits. For this purpose, the sum value of the number of binary variables is first determined. The basis for the calculation is a known QUBO formulation, as well as the specified problem instance. The sum value can in turn be used to estimate the number of (physically) required qubits. We store the calculated sum value in *numVar*, the number of qubits in *numQubits*. Both values are used in the next step.

Step 4. In this step, the appropriate solvers are selected from a set of stored solvers. Their technical data are matched with *numVar* and *numQubits*. QPU solvers have an upper limit in terms of qubits (attribute *maxQubits* in Figure 1). If *numQubits* is larger than the number of actually available qubits, then this QPU solver is not a candidate to solve this problem. Hybrid solvers are limited in the number of variables⁴ (attribute *maxVar* in Figure 1). This value must be higher than *numVar* for a hybrid solver to be considered.

⁴ Default values for *maxVar*: <https://www.dwavesys.com/media/soxph512/hybrid-solvers-for-quadratic-optimization.pdf>

Step 5. In this step, the list of suitable solvers from Step 4 is sorted. The most promising solvers appear first in the sorting. The underlying sorting principle is as follows: If benchmarks exist for certain combinations of problem instances and solvers, the first step is to sort them in descending order according to the quality of the solution. Benchmark scores can be used to determine the quality of the solution. If no benchmarks exist, the results are sorted by solver types and quantum annealers with the highest number of qubits and the most interconnected hardware topology. Assuming that appropriate benchmarks exist, the first step is to sort the solvers using the one-dimensional Euclidean distance according to the following rule:

The following variables and definitions are given:

- $B = (B_1, B_2, \dots, B_m)$, $m \in \mathbb{N}$: Represents the sequence of m benchmark objects available for a specific optimization problem and refers to the problem sizes that were evaluated.
- Each benchmark object B_j contains a fixed number x of integer values that represent possible benchmark scores:

$$B_j = (w_{j1}, w_{j2}, \dots, w_{jx}), j \in \{1, \dots, m\}, x \in \mathbb{N}. \quad (4)$$

Depending on the benchmark, x can be variable and is the maximum number of tested solvers for a benchmark. To ensure a consistent data storage structure, the benchmarks are always sorted in ascending order by the value w_{j1} .

- $w_{j1} \in \mathbb{N}_+$: The value is called the 'main parameter', which represents the reference value being searched for and is fixed. In relation to the benchmarks made for the traveling salesman problem, the value could be the number of nodes in a graph on which the test was carried out [SPA22].
- $n \in \mathbb{N}_+$: Is the value entered by the user for which we are searching for the nearest w_{j1} . Assuming that we are looking specifically at the benchmarks for the traveling salesman problem, the value could be the number of nodes in a graph.
- A function that extracts the first value w_{j1} from a benchmark object B_j :

$$f(B_j) = w_{j1}, \forall j \in \{1, \dots, m\}. \quad (5)$$

- Set of all extracted values:

$$M = (f(B_j) \mid j = 1, \dots, m) = (w_{11}, w_{21}, \dots, w_{m1}). \quad (6)$$

This sequence only contains the first value of each benchmark object. As a restriction, benchmarks are currently only displayed for a single run.

- A function c that defines an ordering relation that compares two elements $(x, y) \in M$ pairwise according to their distance from n .

$$c(x, y) = \begin{cases} -1, & \text{if } |x - n| < |y - n| \\ 0, & \text{if } |x - n| = |y - n| \\ 1, & \text{if } |x - n| > |y - n| \end{cases} \quad (7)$$

where $x \neq y$ for each pair.

- A function $g : \mathbb{N} \rightarrow M$ with:

$$g(c(x, y)) = \begin{cases} x, & \text{if } c(x, y) \leq 0 \\ y, & \text{if } c(x, y) > 0 \end{cases} \quad (8)$$

which returns an x that is closest to n . If several values have the same distance from n , the first value x found is used. If no x is found, the first y that does not deviate from n by 10% is used. Otherwise, the default sort order (as Step 2 below) will be used.

- $S = \{S_1, S_2, \dots, S_u\}$, $u \in \mathbb{N}$: Represents the set of u solver.
- Each solver S_j is described by four characteristics that are relevant for the calculation:
 - $s_j \in \mathbb{N}_0$: *solutionQuality*
 - $v_j \in \mathbb{N}_0$: *maxVariables*
 - $q_j \in \mathbb{N}_0$: *maxQubits*
 - N_j : Solver name.

There are other characteristics that are not initially relevant for the calculation.

The benchmark with the appropriate value is then determined as follows:

$$j^* = \min\{j \mid j \in \{1, \dots, m\}, w_{j1} = x \text{ or } y \text{ (see Eq. 8)}\}, \quad (9)$$

where j^* is the direct identification of the corresponding index of the benchmark. The benchmark object B^* is then determined as follows:

$$B^* = B_{j^*} = (w_{j^*1}, w_{j^*2}, \dots, w_{j^*x}). \quad (10)$$

The *solutionQuality* s_j (i.e., the corresponding benchmark score, cf. Eq. 4) is chosen from B^* according to an assignment rule:

$$s_j = \begin{cases} w_{j^*2}, & \text{if } N_j = \text{"Solver 1"} \\ w_{j^*3}, & \text{if } N_j = \text{"Solver 2"} \\ w_{j^*4}, & \text{if } N_j = \text{"Solver 3"} \\ w_{j^*5}, & \text{if } N_j = \text{"Solver 4"} \\ \vdots & \text{(if there are other names, they will be assigned accordingly)} \end{cases} \quad (11)$$

The set of solvers S^* after assigning the quality of the solution from all solvers is:

$$S^* = \{(S_j, s_j) \mid S_j \in S, s_j \text{ according to the above rule}\} \quad (12)$$

The sorted set is obtained by lexicographic sorting of S^* using the following keys:

$$S_i^* \prec S_j^* \iff (s_i < s_j) \text{ or } (s_i = s_j \text{ and } v_i < v_j) \\ \text{or } (s_i = s_j \text{ and } v_i = v_j \text{ and } q_i < q_j) \quad (13)$$

The sorted set S^{**} is then given by:

$$S^{**} = (S^*, \prec). \quad (14)$$

As a second step, if there are no benchmarks, the solvers are sorted according to the following rule. Let the definitions of solvers S , $\max Variables\ v_j$ and $\max Qubits\ q_j$ remain unchanged. Then the sorted set is obtained by the following hierarchical sorting:

$$S_i \prec S_j \iff (v_i < v_j) \text{ or } (v_i = v_j \text{ and } q_i < q_j). \quad (15)$$

The sorted set S^* is then given by:

$$S^* = (S, \prec). \quad (16)$$

Step 6. In the last step, it is the user's turn again. The user is informed of the estimated number of qubits required and receives a sorted list of solvers. More information on the solvers (such as prices) is given if available.

5 Implementation and Evaluation Method

In this section, we demonstrate how the prototype works using the TSP and describe the evaluation method. In doing so, we will go through the individual steps shown in Figure 1. Figures 2-A and 2-B show the QC-Adviser dialogs for user interactions.

Step 1. A user is interested in solving a TSP with a quantum annealer and would like a recommendation in this regard. To do so, the user selects TSP (cf. Figure 2-A).

| | | | |
|--|---|--|--|
| Routing Problems Sequencing Problems A General Problems | | Routing Problems Sequencing Problems B General Problems | |
| In a general routing problem, an attempt is made to determine... | | In a general routing problem, an attempt is made to determine... | |
| Please select one <input checked="" type="checkbox"/> General Route Optimization (TSP) Vehicle Routing Optimization Capacitated Vehicle Routing Problem with Time and State Multi-Depot Capacitated Vehicle Routing Shipment Rerouting Robot Trajectory Planning | | General Route Optimization (TSP) | |
| | | Nodes * 4 | |
| Quantum service provider | Solver | Benchmark score | C Estimation |
| D-Wave Leap-Ocean | Name: hybrid_binary_quadratic_model_version2 Technology: Hybrid Solver: BQM Topology: Pegasus | 100 % success rate | Est. no. of qubits: 112 Est. no. of variables: 16 |
| D-Wave Leap-Ocean | Name: Advantage_system4.1 Technology: QPU Topology: Pegasus | 100 % success rate | Est. no. of qubits: 112 Est. no. of variables: 16 |

Fig. 2: Compilation of the QC-Adviser user interface sections for workflow step 1 (Subsection A: Problem selection), workflow step 2 (Subsection B: Specification of the problem instance), and workflow step 6 (Subsection C: List of solvers found).

Step 2. The user specifies the TSP problem instance. The user is interested in the basic form of the problem, which corresponds to a complete graph (each pair of nodes is connected by an edge). Therefore, it is sufficient to specify only the number of nodes. The user decides for 4 nodes (cf. Figure 2-B). Note: Further development of the prototype will allow for the formulation of more complex TSP problems, including those that involve distance specifications, such as in the work of Stogiannos et al. [SPA22].

Steps 3-5. In these steps, the internal calculation of the QC-Adviser takes place without user interaction. The basis for computing *numVar* and *numQubits* for the TSP is based on the QUBO formulation mentioned in Section 2. From the QUBO formulation, the formula n^2 for calculating *numVar* was derived [SPA22], where n is the number of nodes specified in Step 2. Based on this, the number of qubits required is estimated, and the corresponding solvers are selected and sorted.

Step 6. The solvers are presented to the user in tabular form (cf. Figure 2-C). The solver information, the estimated number of qubits required, and the benchmark score are displayed. The latter is only displayed if a benchmark score is available for the combination of problem instance and solver. The value indicates how often this solver correctly calculated the expected result (for a given number of runs). The user also has the option of calling up the usage prices.

Some implementation aspects. The prototype offers support for various problems (currently 15) and is not tailored to the TSP. The different QUBO formulations and the estimation approaches implemented come from the research literature. As examples, we mention the following research literature: For the TSP and the Vehicle Routing Problem the work of [SPA22] and [Bo20], for Job Shop Scheduling the work of [CDC22] as well as the work of [Lu14], which contains a good overview of different problems.

The prototype also has a modular structure and follows the MVC paradigm. This means that developers can add new problem classes, problems, and individual hardware datasets (for solvers) without affecting the overall application. The datasets are inserted in the form of JSON documents and form the *QC-Services collection* in Figure 1. Fast-changing information, such as prices, can be obtained automatically via the service provider interface, if available.

Evaluation method. We outline the evaluation method using a problem for which a benchmark is available. If there are no benchmarks, the sorting is performed according to Eq. 15.

1. We define a problem instance (including conditions) and determine its best possible solution using classical methods. Such a solution is either known, obvious, or can be calculated in a reasonable amount of time if the problem instance is not too large.
2. We enter this problem instance into the QC-Adviser to obtain the sorted solvers.
3. Solution calculations for the given problem instance with at least two different solvers: To do this, we select two solvers. The first solver is at the top of the list, the second

behind it, or at the end of the list. We calculate the solution to the problem instance using both solvers.

4. Comparison of the best possible classical solution with the solutions determined by the solvers in order to evaluate the QC-Adviser (i.e., to ensure the correctness of the recommended solvers): Assume that the quality of the solution (benchmark score) of the first solver is 100%. Then its solution should correspond to the classical best result (otherwise this solver would be incorrectly sorted because the top solvers deliver the best solutions). And assuming that the quality of the solution of the second solver is significantly below 100%. Then the result of this solver must deviate from the classically determined best result by a certain percentage (the further down a solver is in the list, the greater the deviation from the optimal result). In the case of TSP, this would mean that the determined tour takes longer than the optimally determined tour (i.e., a percentage deviation can be detected).
5. In other problems, a different metric must be used, such as the makespan in the job shop scheduling problem. The overall evaluation is based on a systematic comparison of a classical problem solution with the solutions provided by the solvers for each problem, as well as the positions of the solvers.

6 Conclusion

We have introduced the QC-Adviser prototype, a dialogue-based tool that supports users in selecting suitable solvers. Users only make domain-specific input and do not need any quantum computing know-how. The QC-Adviser results view lists the recommended solvers with basic information in a sorted list. The evaluation to date has been literature-based and selective. For a given problem instance and solution quality (benchmark scores) from at least two different solvers, we can check the expected positions of these solvers in the QC-Adviser list, as the solvers are sorted in descending order according to solution quality. The QC-Adviser tool is publicly available at Zenodo⁵.

A comprehensive evaluation using the method described in Section 5 (with experiments conducted on real quantum hardware) will be considered in future work. In addition, the prototype can be further developed. More detailed problem-specific forms could be developed for the specification of the constraints. Additional solvers such as D-Wave's *nonlinear solver* may also be considered.

References

- [Aw23] Awasthi, A. et al.: Quantum Computing Techniques for Multi-knapsack Problems. In: Intelligent Computing. Springer Nature Switzerland, pp. 264–284, 2023.
- [Ba21] Bayerstadler, A. et al.: Industry quantum computing applications. EPJ Quantum Technology 8 (1), p. 25, 2021.

⁵ <https://zenodo.org/records/14513037>

-
- [Bo20] Borowski, M. et al.: New Hybrid Quantum Annealing Algorithms for Solving Vehicle Routing Problem. In: ICCS (6). Vol. 12142. Lecture Notes in Computer Science, pp. 546–561, 2020.
 - [CDC22] Carugno, C.; Dacrema, M. F.; Cremonesi, P.: Evaluating the job shop scheduling problem on a D-wave quantum annealer. *Scientific Reports* 12 (1), 2022.
 - [Gl22] Glover, F. W. et al.: Quantum bridge analytics I: a tutorial on formulating and using QUBO models. *Ann. Oper. Res.* 314 (1), pp. 141–183, 2022.
 - [HC23] How, M.-L.; Cheah, S.-M.: Business Renaissance: Opportunities and Challenges at the Dawn of the Quantum Computing Era. *Businesses* 3 (4), pp. 585–605, 2023.
 - [Lo25] Lorenz, J. M. et al.: Systematic benchmarking of quantum computers: status and recommendations, 2025.
 - [Lu14] Lucas, A.: Ising formulations of many NP problems. *Frontiers in Physics* 2, 2014.
 - [Po23] Poggel, B. et al.: Recommending Solution Paths for Solving Optimization Problems with Quantum Computing. In: QSW. IEEE, pp. 60–67, 2023.
 - [QBW25] Quetschlich, N.; Burgholzer, L.; Wille, R.: MQT Predictor: Automatic Device Selection with Device-Specific Circuit Compilation for Quantum Computing. *ACM Transactions on Quantum Computing* 6 (1), 2025.
 - [Sa20] Salm, M. et al.: The NISQ Analyzer: Automating the Selection of Quantum Computers for Quantum Algorithms. In: SummerSOC. Vol. 1310. Communications in Computer and Information Science, pp. 66–85, 2020.
 - [Sa23] Salm, M. et al.: How to Select Quantum Compilers and Quantum Computers Before Compilation. In: CLOSER. Pp. 172–183, 2023.
 - [Se24] Sehrawat, V.: Quantum Computing: Algorithms and Applications in Optimization Problems. *Journal of Quantum Science and Technology* 1 (2), pp. 18–22, 2024.
 - [Si24] Singh, P. et al.: A Survey on Available Tools and Technologies Enabling Quantum Computing. *IEEE Access* 12, pp. 57974–57991, 2024.
 - [SPA22] Stogiannos, E.; Papalitsas, C.; Andronikos, T.: Experimental Analysis of Quantum Annealers and Hybrid Solvers Using Benchmark Optimization Problems. *Mathematics* 10 (8), p. 1294, 2022.
 - [TTK25] Tripathi, R.; Tomar, S.; Kumar, S.: A Comprehensive Survey on Quantum Annealing: Applications, Challenges, and Future Research Directions. 2025.
 - [We20] Weder, B. et al.: The Quantum software lifecycle. In: APEQES@ESEC/SIGSOFT FSE. Pp. 2–9, 2020.
 - [Ya22] Yarkoni, S. et al.: Quantum annealing for industry applications: Introduction and review. *Reports on Progress in Physics* 85 (10), p. 104001, 2022.
 - [ZRS24] Zajac, M.; Restat, V.; Störl, U.: Quantum versus Classical Computation: Automatic Decision-Making Approaches. In: INFORMATIK. Vol. P-352. LNI, pp. 573–577, 2024.