Appendix A Selected articles

This appendix contains the table that summons the identification numbers, titles, and references of the selected articles.

Table 5 Selected articles.

10	Multiclass Land Use / Land Cover (LULC) Classification Using Quantum Enhanced Support Vector Machines	[60]
12	Effect of Pure Dephasing Quantum Noise in the Quantum Search Algorithm Using Atos Quantum Assembly	[16]
16	QuantoTrace: Quantum Error Correction as a Service for Robust Quantum Computing	[55]
39	Comprehensive Library of Variational LSE Solvers	[17]
41	Comparing Natural Language Processing and Quantum Natural Processing approaches in text classification tasks	[24]
50	Introduction to Quantum-Train Toolkit	[49]
61	Composable Quantum Oracles for Shifting Quantum Circuits Abstraction Level	[40]
66	Quantum-Classical-Quantum Workflow in Quantum-HPC Middleware with GPU Acceleration	[11]
69	Polynomial Reduction Methods and their Impact on QAOA Circuits	[73]
72	Quantum Program Testing Through Commuting Pauli Strings on IBM's Quantum Computers	[56]
77	Evolution of Service-Oriented Computing: Integrating Quantum Techniques for Integer Factorization	[61]
78	Hybrid quantum architecture for smart city security	[47]
99	Quantum Metropolis Solver: a quantum walks approach to optimization problems	[50]
105	Operating with Quantum Integers: An Efficient 'Multiples of' Oracle	[41]
108	Improving the Quality of Quantum Services Generation Process: Controlling Errors and Noise	[76]
115	Quantum Computing Techniques for Multi-knapsack Problems	[72]
117	Lattice surgery-based logical operations in a fault-tolerant quantum software framework	[25]
122	A Reference Implementation for a Quantum Message Passing Interface	[2]
134	Quantum Computing with Differentiable Quantum Transforms	[53]
135	ScaffML: A Quantum Behavioral Interface Specification Language for Scaffold	[36]
141	Chaotic Image Encryption Based on Boson Sampling	[37]
165	Evaluation of Entaglement-based Quantum Key Distribution for Genome Data Transmission	[31]
170	Solving 2 by 2 Grid Sudoku Problem using Grover's Algorithm with Intel Quantum SDK	[62]
192	QUANTUM ANNEALING LEARNING SEARCH IMPLEMENTATIONS	[12]
199	RECONSTRUCTING BAYESIAN NETWORKS ON A QUANTUM ANNEALER	[14]
230	Secure Software Leasing Without Assumptions	[1]
256	Password authentication schemes on a quantum computer	[3]
271	Variational quantum chemistry programs in jaqalpaq	[6]
289	Procedural generation using quantum computation	[43]
294	A quantum procedure for map generation	[44]
295	Quantum algorithms for near-term devices	[38]
297	Quantum Software Engineering Supremacy in Intelligent Robotics	[68]
311	TIGER: Topology-aware Task assignment approach using ising machines	[35]

Appendix B Tables

This appendix contains the tables that summon the number of times each of the terms appeared in the six different research questions as well as the categorisation, if any, of terms within each research question.

Table 6 Quantum algorithms, or software solutions mentioned or proposed and the number of times each one of them appeared (counts) in the 33 primary studies for RQ1.

Quantum algorithm, or solution proposed	Counts
Grover	5
VQE	4
QAOA	3
Quantum Annealing	3
QUBO	2
QML	2
Shor	1
Metropolis-Hastings	1
Quantum Walks	1
Boson Sampling	1
QOPS	1
QKD	1

Table 7 Quantum simulators or simulation platforms employed, related to RQ2.

Quantum simulator/simulation platform	Times employed
IBM Quantum	9
PennyLane	4
DWave	2
HPC Simulators	1
AQASM	1
myQLM	1
Quantum Learning Machine (QLM)	1
Jax library	1
Cirq	1
cuQuantum SDK	1
Amazon Braket	1
QPlayer	1
QASMBench	1
StrawberryFields	1
Intel Quantum SDK	1

Table 8 Real quantum computer platform employed, related to RQ2.

Real quantum computing platform	Times employed
IBM Quantum	8
DWave	6
Amazon Braket	1
Xanadu Quantum	1
QSCOUT	1

Table 9 Parameters or other criteria to evaluate quantum software and the number of times each one of them appeared (counts) in the 30 primary studies for RQ3.

Term/parameter/criteria	Counts
Performance	12
Speedup	12
Number of qubits	12
Complexity	12
Accuracy	9
Depth	9
Efficiency	7
Quality	7
Probability	7
Effectiveness	5
Runtime	4
Scalability	4
Reusability	4
Shots	4
Execution time	3
Stability	3
Fidelity	3
Shallow (depth)	3
Number of gates	3
Computational complexity	3
Cost function	3
Computational speed	2
Computational cost	2
Problem size	2
Success rate	2
Composability	2
Robustness	2
Computational time	1
Hardware efficiency	1
Time-to-solution	1
Precision	1
Reliability	1
Memory consumption	1
Cost-effective	1
Accessibility	1
Expressivity	1
Loss function	1
Learning rate	1
Training time	1
Success probability	1
Solution probability	1
Error probability	1
Quantum annealing sensitivity	1
Streamlined	1
Controllability	1
Suitability	1
Communication cost	1
Counts	1

Quantum Utility	1

Table 10 Responses for RQ3 categorized as performance and efficiency, result accuracy and success, resources and complexity, implementation, and others.

Performance and efficiency			
Performance	Efficiency	Speedup	Computational speed
Execution time	Runtime	Computational time	Computational cost
Time-to-solution		Effectiveness	
Result accuracy and success			
Accuracy	Precision	Reliability	Quality
Stability	Fidelity	Success probability	Probability
Success rate	Solution probability	Error probability	
Resources and complexity			
Number of qubits	Depth	Shallow (depth)	Number of gates
Problem size	Computational complexity	Complexity	Memory consumption
Cost-effective	Scalability	Accessibility	Expressivity
Cost function	Loss function	Learning rate	Training time
Implementation			
Controllability	Reusability	Composability	Streamlined
Quantum annealing sensitivity		Robustness	
Other terms			
Suitability	Communication cost	Shots	Counts
Quantum Utility			

Table 11 Verification or validation of quantum software and the number of times each one of them appeared (counts) in the 13 primary studies for RQ4.

Quantum verification or validation	**Counts**
Accuracy	4
Effectiveness	2
Simulation testing	2
Success rate	1
Success probability	1
Accuracy rate	1
Model accuracy	1
Precision	1
Recall	1
F1-Score	1
Probability	1
Test assessment	1
Benchmark: a simplified game based on the work of Broadbent et al. on Secure Software Leasing (SSL)	1
Benchmark: inverted pendulum "Cart-Pole" problem	1
Usage of a quantum annealer to check	1
Standard error	1
Evaluation of the sum of all penalty terms in the QUBO	1
Verification of data authenticity	1

ANSI/ISO C Specification Language (ACSL)	1
Quantum probability computation	1
Quantum Utility	1
Benchmark: N-Queens	1
Analysis of Quantum Message Authentication Codes (QMACs)	1
Honest-Malicious security model	1
Experimentation and comparative analysis with classical approaches	1

Table 12 Limitations of quantum software and the number of times each one of them appeared (counts) in the 26 primary studies for RQ5.

Quantum software limitations	**Counts**
Number of qubits	12
NISQ era	12
Error	11
Noise	10
Decoherence/coherence time	8
Qubit connectivity	5
Size	5
Scalability	4
Gate related (imperfections, inaccuracies or errors)	3
Fault-tolerancy	3
Readout	2
Limited machine time	2
Intermediate verification of results is not possible	2
Lack of specialized expertise	2
Slow execution time	2
Measurement errors	1
State preparation	1
Efficiency in applying gates	1
Limited RAM of simulators	1
Usage queues	1
Qubit fidelity	1
Explicit specification of expected outputs	1
Generating random test cases is time-consuming and resource-intensive	1
The invocation of quantum services is technically difficult	1
Quantum algorithms and their parameters are architecture-dependent	1
Quantum computers are expensive to produce and operate	1
Lack of do-it-yourself algorithms	1
Infrastructure dependence	1
Current abstractions used to define quantum service architectures	1
Reliability	1
Uncertainty in the outputs	1
Usability	1
Gate types	1
Embedding overhead	1
The noise model of the computer does not reflect the actual amount of noise	1
Noise mitigation	1
The quantum no-cloning theorem makes it impossible to copy and paste quantum states	1
Lack of Behavioral Interface Specification Languages	1

Problem mapping	1
Find a suitable way to represent the objective function (annealing)	1
Structure of the annealer topologies prevents a straightforward representation of the problem	1
Internal system constraints impose a maximum annealing time	1
Programming complexity	1
Variable range	1
Qubit weights and coupler strengths	1
Relaxation time	1
The need for multiple qubits to encode input data limits the complexity of the datasets	1
Non intentional interactions between qubits	1

Table 13 Responses for RQ5 categorized as hardware, operational, development, and infraestructure.

Hardware limitations			
Number of qubits	Noise	Decoherence time	Error
Gate types	Measurement errors	NISQ era	Fault-tolerancy
Qubit connectivity	Scalability	Readout	State preparation
Efficiency in applying gates	Qubit fidelity	Size	Reliability
Uncertainty in the outputs	Gate related (imperfections, inaccuracies or errors)		
The noise model of the computer does not reflect the actual amount of noise			Noise mitigation
Relaxation time	Qubit weights and coupler strengths	Non intentional interactions between qubits	
The need for multiple qubits to encode input data limits the complexity of the datasets			
The quantum no-cloning theorem makes it impossible to copy/paste quantum states			
Operational limitations			
Limited machine time	Slow execution time	Usage queues	Limited RAM of simulators
Internal system constraints impose a maximum annealing time			
Development limitations			
Lack of do-it-yourself algorithms	Variable range	Problem mapping	Embedding overhead
Explicit specification of expected outputs		Intermediate verification of results is not possible	
Lack of Behavioral Interface Specification Languages		Programming complexity	
Generating random test cases is time-consuming and resource-intensive			
Find a suitable way to represent the objective function (annealing)			
Structure of the annealer topologies prevents a straightforward representation of the problem			
Quantum algorithms and their parameters are architecture-dependent			
Infraestructure limitations			
Lack of specialized expertise		Usability	Infraestructure dependence
Current abstractions used to define quantum service architectures			
The invocation of quantum services is technically difficult			
Quantum computers are expensive to produce and operate			

Table 14 Quantum software challenges found in the 14 primary studies for RQ6.

Quantum software challenges
Quantum computing (QC) is still in its infancy
Performance on NISQ hardware will be very low
Developing devices that work according to the laws of quantum mechanics
Developing fault-tolerant quantum error correction codes
Errors arising from factors like noise, defective hardware, and decoherence

Maintaining the integrity of quantum information
Reliability and fidelity of quantum computations
Achieving high-fidelity error correction
Quantum devices have been available only since the past few years
Devices are still imperfect
The need for multiple qubits to encode input data adds computational cost and complexity
The industry will not adopt quantum devices unless quantum software can be produced in a repeatable, efficient, maintainable, reusable, and cost-effective way
Better documentation about how to reuse quantum circuits
Quantum interconnect bottleneck (QIB)
Adoption of distributed quantum algorithms introduces vulnerabilities
Intensive computational requirement of Quantum Machine Learning (QML)
Enhancing accuracy, extending applicability
Higher-level process of formulating a real-world problem and transforming it to a suitable representation for quantum computers
High-level software engineering, low-level design of compilers
Current software testing methods are not designed to work efficiently with error mitigation techniques
Challenges using real quantum computers like IBM: incompatible test cases, requirement of a complete program specification
Incompatibility with error mitigation methods
Detecting phase flip faults
Scalability and practicality of current testing methods in real-world scenarios
The invocation of a quantum program in an agnostic manner
Identification of new quantum algorithms for integer factorization
Reworking and extending classical software engineering into the quantum domain
Qubits are inherently fragile and susceptible to decoherence
Qubits can exist in a linear combination of both 0 and 1 states simultaneously
Low abstraction level, absence of integration, deployment, or quality and security control mechanisms
Errors pose challenges for addressing quality and security requirements
Lack of support tools for quantum software design
Simulating general-purpose quantum circuits
Realization of individual quantum processors with millions of physical qubits
Achieving fault tolerance in quantum gates
Transforming a problem into QUBO format

Table 15 Responses for RQ6 categorized as hardware, error correction, integration, problem formulation and representation, scalability, and adoption and usage.

<table>
<tr><th colspan="3">Hardware challenges</th></tr>
<tr><td colspan="2">Quantum computing (QC) is still in its infancy</td><td>Performance on NISQ hardware will be very low</td></tr>
<tr><td>Devices are still imperfect</td><td colspan="2">Quantum devices have been available only since the past few years</td></tr>
<tr><td colspan="2">Developing devices that work according to the laws of quantum mechanics</td><td>Qubits are inherently fragile and susceptible to decoherence</td></tr>
<tr><td colspan="2">Realization of individual quantum processors with millions of physical qubits</td><td>Achieving fault tolerance in quantum gates</td></tr>
<tr><th colspan="3">Error correction challenges</th></tr>
<tr><td colspan="2">Developing fault-tolerant quantum error correction codes</td><td>Detecting phase flip faults</td></tr>
<tr><td colspan="2">Maintaining the integrity of quantum information</td><td>Reliability and fidelity of quantum computations</td></tr>
<tr><td colspan="2">Achieving high-fidelity error correction</td><td>Incompatibility with error mitigation methods</td></tr>
<tr><td colspan="3">Errors arising from factors like noise, defective hardware, and decoherence</td></tr>
</table>

Integration challenges
Better documentation about how to reuse quantum circuits
Low abstraction level, absence of integration, deployment, or quality and security control mechanisms
Errors pose challenges for addressing quality and security requirements
Lack of support tools for quantum software design
Simulating general-purpose quantum circuits
Reworking and extending classical software engineering into the quantum domain
High-level software engineering, low-level design of compilers
The invocation of a quantum program in an agnostic manner
Problem formulation and representation challenges
The need for multiple qubits to encode input data adds computational cost and complexity
Intensive computational requirement of Quantum Machine Learning (QML)
Enhancing accuracy, extending applicability
Higher-level process of formulating a real-world problem and transforming it to a suitable representation for quantum computers
Transforming a problem into QUBO format
Identification of new quantum algorithms for integer factorization
Scalability challenges
Scalability and practicality of current testing methods in real-world scenarios
Current software testing methods are not designed to work efficiently with error mitigation techniques
Challenges using real quantum computers like IBM: incompatible test cases, requirement of a complete program specification
Qubits can exist in a linear combination of both 0 and 1 states simultaneously
Adoption and usage challenges
The industry will not adopt quantum devices unless quantum software can be produced in a repeatable, efficient, maintainable, reusable, and cost-effective way
Quantum interconnect bottleneck (QIB)
Adoption of distributed quantum algorithms introduces vulnerabilities