1 QuantumShare Knowledge-Based System

This chapter describes the implementation of QuantumShare into a functioning knowledge-based system. As a result, the implementation facilitates end usage of QuantumShare. Additionally, the implementation enables testing through writing and running SPARQL queries based on the competency questions (i.e., functional requirements). Hence, this chapter contributes to answering SRQ 7.

First, Section 7.1 denotes the functional architecture of the system. Subsequently, the technical implementation that aims to satisfy the functional design is described in Section 7.2. Lastly, Section 7.3 describes the evaluation of the knowledge-based system through use cases, competency questions, and SPARQL queries.

1.1 Functional Architecture

The functional architecture provides a high-level workflow model of the system's components. The architecture, as seen in Figure 29, represents the component's functions and interactions.

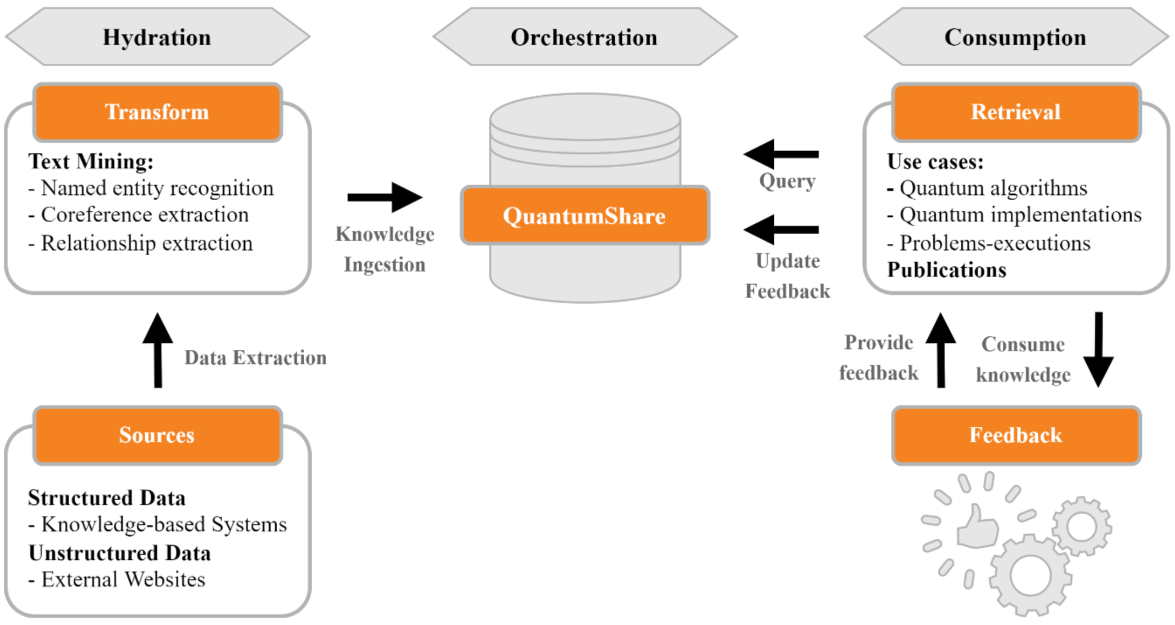


Figure 1, Functional architecture of the QuantumShare knowledge-based system, inspired by (Jonge & Satyanarayana, 2021)

The knowledge-based system, which is inspired by the fieldnotes from Jonge & Satyanarayana (2021), involves three major processes:

Hydration

Firstly, hydration refers to extracting and pre-processing relevant data to ensure data is ingested in the knowledge base in a standardized format. This process refers mainly to the knowledge acquisition process of Fensel et al. (2020), explained in Section 2.2. It involves extracting the data from various sources, such as existing knowledge bases (i.e., structured data) or text on external websites (i.e., unstructured data). Subsequently, the raw data requires a systematic transformation to align the data format with knowledge present in the knowledge base. In this context, data transformation consists of named entity recognition (i.e., classifying knowledge in unstructured text to the defined entities), coreference extraction (i.e., different expressions refer to a similar concept), and relationship extraction (i.e., detecting semantic relationships in text).

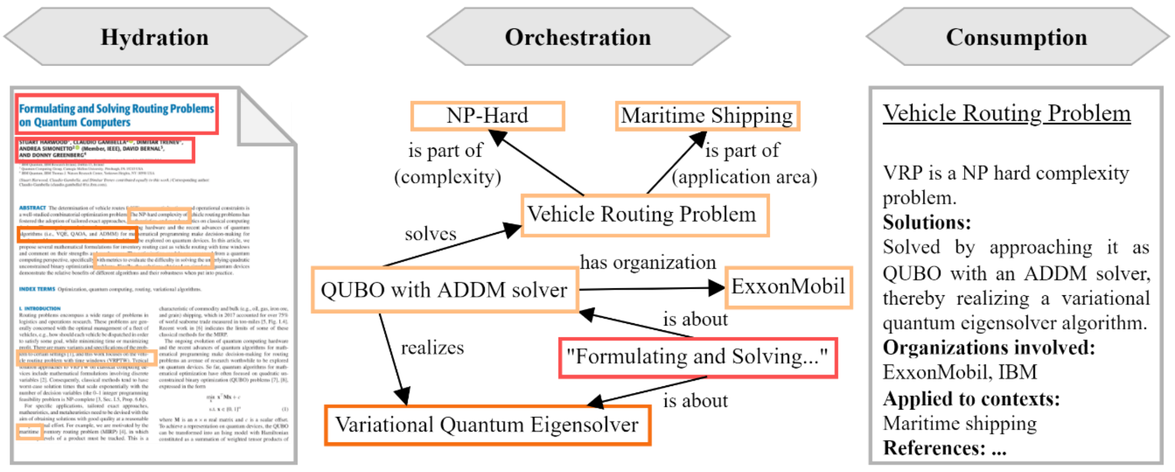
Orchestration

Afterward, the ingested data is organized and merged with existing knowledge captured in the system, referring to the orchestration process. Orchestration is similar to the knowledge representation process explained in Section 2.2. Finally, graphs that structurally represent the data are constructed, including existing entities, relationships, and semantic descriptions present in the QuantumShare ontology (Ji et al., 2021). This way, the graph visualizes the data stored in the knowledge base, which are triples (e.g., *Superpolynomial, rdfs:subClassOf, Speedup_Class*) based on RDF.

Consumption

Lastly, consumption refers to the client-side interaction with the knowledge base. The user should be able to retrieve knowledge about use cases (i.e., algorithms, implementations, and problem-executions) and publications that refer to those use cases. By enabling the formulation of queries, the knowledge base can be leveraged. In addition, end-users may not agree with the knowledge provided by the system, either through the formulation of queries or the semantic relationships defined. By facilitating a feedback mechanism, the knowledge base can be updated accordingly.

Figure 30 illustrates different representations of information throughout the three major processes (i.e., hydration, orchestration, and consumption) in an accessible way. After extracting a relevant scientific publication on quantum computing, processes indicate an example of the information flow. Firstly, the hydration process classifies the content of this publication into defined entities (e.g., its title and authors, problem complexity, and organizations involved). For example, it classifies "*Formulation and Solving Routing Problems on Quantum Computers*" as the title present in the metadata of the publication. Next, the identified knowledge is represented by leveraging the semantic relationships defined in the ontological model. For instance, Figure 30 represents the "*Vehicle Routing Problem*" as part of the application area "*Maritime Shipping*" and complexity class "*NP-Hard*". Finally, the knowledge is presented to an end-user by assembling the facts resulting from a user information request. Although Figure 30 illustrates this step, assembling the knowledge in a front-end is considered beyond the scope of this thesis.



Vehicle Routing Problem

VRP is a NP hard complexity problem.
Solutions:
Solved by approaching it as QUBO with an ADDM solver, thereby realizing a variational quantum eigensolver algorithm.
Organizations involved:
ExxonMobil, IBM
Applied to contexts:
Maritime shipping
References: ...

Figure 2, Conceptual view of information flow in QuantumShare processes: hydration, orchestration, and consumption

1.2 Implementation

Various cloud services offered by Amazon Web Services (AWS) are implemented to satisfy the functional design described in the previous chapter. The technical architecture illustrated in Figure 31 aims to provide a high-level view of the interaction between these cloud services. Therefore, the illustration is organized with the same processes identified in the functional architecture (Figure 29) to ease the understanding of the implementation. However, as can be deducted from Figure 31, the feedback mechanism is not implemented due to the time constraints related to this thesis research. In addition, the hydration process, more specifically the transformation component, is carried out semi-automatically.

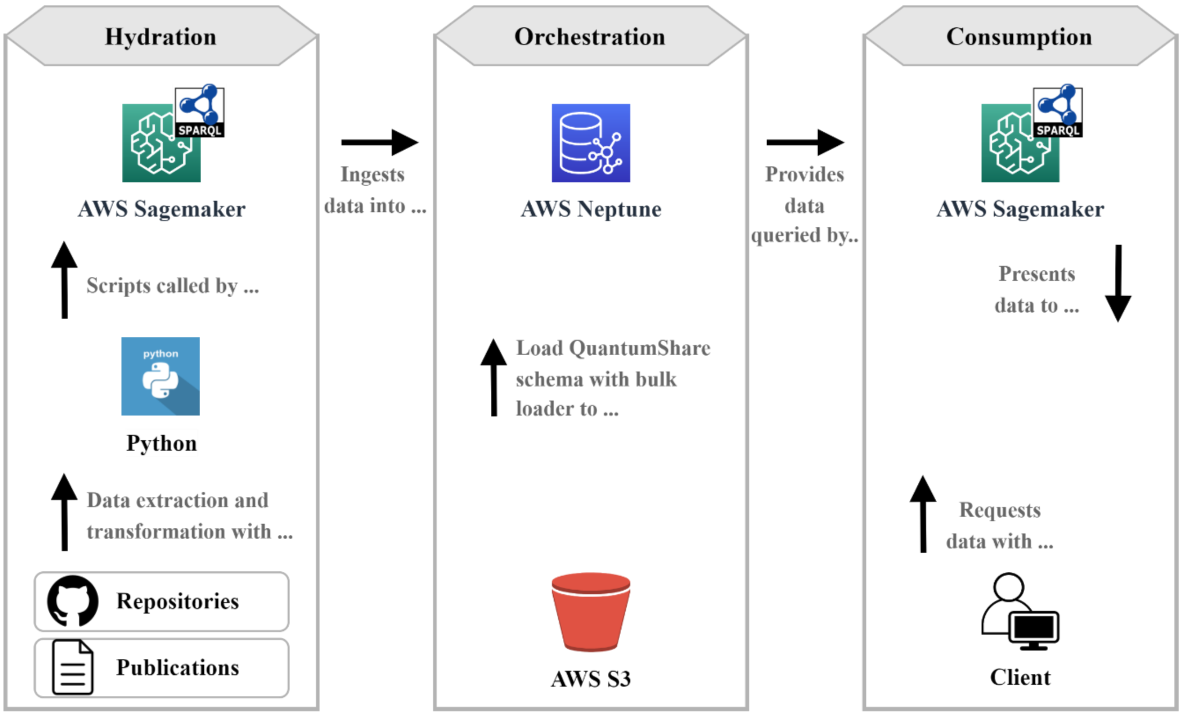


Figure 3, Technical architecture of the QuantumShare knowledge-based system

QuantumShare Schema Upload (Orchestration)

The QuantumShare knowledge base is deployed on Amazon Neptune, a fully-managed graph database service optimized for storing and querying graph data (Amazon Web Services, 2018). Firstly, the QuantumShare schema, elaborated in Chapter 6, was exported from ontology editor Protégé in Turtle syntax and subsequently stored in an Amazon Simple Storage Service (S3) bucket. Subsequently, Neptune's bulk loader command was used to load the QuantumShare schema directly into a created Neptune database cluster, as the loader is optimized for large datasets.

Data Extraction, Transformation, and Ingestion (Hydration)

Two types of data sources were used to populate the ontology with instances. Firstly, GitHub repositories served as a data source for relatively structured information, often in JSON format. For example, data was extracted from the repository of the Quantum Algorithm Zoo (QAZ), a catalog maintained by Jordan (2021). In total, 64 quantum algorithms, including their algorithm class, speedup class, and description, were retrieved from QAZ. Furthermore, 541 references from these algorithms to scientific publications were extracted, including various metadata. Secondly, additional publications were manually analyzed to capture sample data for the quantum implementation and problem-executed related classes.

Python scripts were created to extract the data from GitHub repositories and transform the data into the required triple pattern format for data ingestion. Triple patterns consist of three entities (i.e., subject, predicate, and object) that capture semantic relationships of knowledge, thereby representing a machine-readable format. For example, the “factoring algorithm is part of algebraic and number-theoretic algorithms” consists of a subject (i.e., factoring algorithm), subject (i.e., is part of), and predicate (i.e., algebraic and number-theoretic algorithms). Besides classifying the subjects, predicates, and objects, preprocessing was carried out to solve any inconveniences related to punctuation within entities.

A Jupyter notebook was hosted in AWS SageMaker, connected to the endpoint of the AWS Neptune database cluster. Next, the Python scripts were imported into the AWS SageMaker environment and invoked from the Jupyter notebook, making the extracted triples accessible. Then, the data was ingested using the “INSERT DATA” operation of graph query language SPARQL. Lastly, the SPARQL queries were submitted to the database by using SPARQLWrapper, a Python package for enabling remote execution of queries to database endpoints (Herman et al., 2014).

Data Retrieval (Consumption)

Various SPARQL queries were written to retrieve the knowledge stored in the Neptune database cluster. These SPARQL queries are based on the competency questions stated in Section 5.3, as these questions represent the information demand of potential end-users. Subsequently, the SPARQL queries were submitted to a dedicated Jupyter notebook in the AWS SageMaker environment. As aforementioned, this environment is connected to the endpoint of the AWS Neptune database cluster, enabling the execution of these SPARQL queries. As a result, the retrieved data is provided in either table or graph format. Several results of the data retrieval process can be seen in Section 7.3.

1.3 Testing through Use Cases, Competency Questions, and Queries

A final assessment of the QuantumShare knowledge base was conducted to indicate the utility and completeness. This section describes several use cases of QuantumShare, formulated as scenarios with corresponding competency questions and SPARQL queries. As proposed by the comprehensibility principle, covered in the non-functional requirements (Section 5.1.2), a potential front-end of QuantumShare should distinguish applications, algorithms, and implementations, as this three-layered approach allows various users (e.g., quantum computing experts and non-expert adopters) to explore QuantumShare’s contents to their interest. Aligned with this intention, the following three use cases are drafted to test the system.

An Exploration of Quantum Algorithms by Experts

As stated in the introduction of this study, creating awareness and understanding the knowledge in scientific publications is of importance to academics as well as professionals to determine room for innovation (Nasar et al., 2018). A potential end-user is an algorithm developer in mathematics and computer science.

Table 1, Competency questions mapped to SPARQL queries for ontology testing

CQ	SPARQL query

6, 8

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX quantumshare: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumShare#>
PREFIX QuantumAlgorithm: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumAlgorithm#>
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>

CONSTRUCT {?algorithm dul:isPartof ?classes .
?algorithm rdfs:subClassOf ?other}

WHERE {?algorithm rdfs:subClassOf* QuantumAlgorithm:Quantum_Algorithm .
?algorithm rdfs:subClassOf ?other .
?algorithm dul:isPartOf ?classes }
```

7, 8

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX quantumshare: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumShare#>
PREFIX QuantumAlgorithm: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumAlgorithm#>
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>

CONSTRUCT {?algorithm dul:isPartof ?classes .
?algorithm rdfs:comment ?description}

WHERE {?algorithm dul:isPartOf ?classes .
?classes rdfs:subClassOf QuantumAlgorithm:Speedup_Class .
?algorithm dul:isPartOf quantumshare:Approximation_and_Simulation_Algorithms .
?algorithm rdfs:comment ?description}
```

30, 33

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX quantumshare: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumShare#>
PREFIX QuantumAlgorithm: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumAlgorithm#>
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>

CONSTRUCT {?ref dul:isAbout ?algorithm .
?ref dul:hasParameter ?metadata.
?metadata dul:hasParameterDataValue ?value}
WHERE {?algorithm rdfs:label "Quantum Approximate Optimization" .
?ref dul:isAbout ?algorithm .
?ref dul:hasParameter ?metadata.
?metadata dul:hasParameterDataValue ?value}
```

Firstly, the algorithm developer may be interested in retrieving all the quantum algorithms captured in QuantumShare, including their corresponding speedup class and overarching algorithm class. This task corresponds with competency questions 6 and 8, answered to the corresponding SPARQL query stated in Table 16. Figure 32 visualizes the graph constructed by the SPARQL query.

Seeing the number of algorithms and the variety of algorithm classes, the algorithm developer decides to focus on his specialization: approximation and simulation algorithms. Furthermore, as the developer prefers a more comprehensive view, they demand a description and speedup class of the algorithm, which concerns competency questions 7 and 8. Figure 33 visualizes the approximation and simulation algorithms and their speedup class and description, resulting from performing the corresponding SPARQL query stated in Table 16. These algorithms achieve three different speedup classes: polynomial for the simulated annealing algorithm (left-hand side), polynomial with exceptions for the semidefinite programming algorithm (middle), and super-polynomial for the remaining algorithms, such as approximate quantum optimization and quantum simulation (right-hand side).

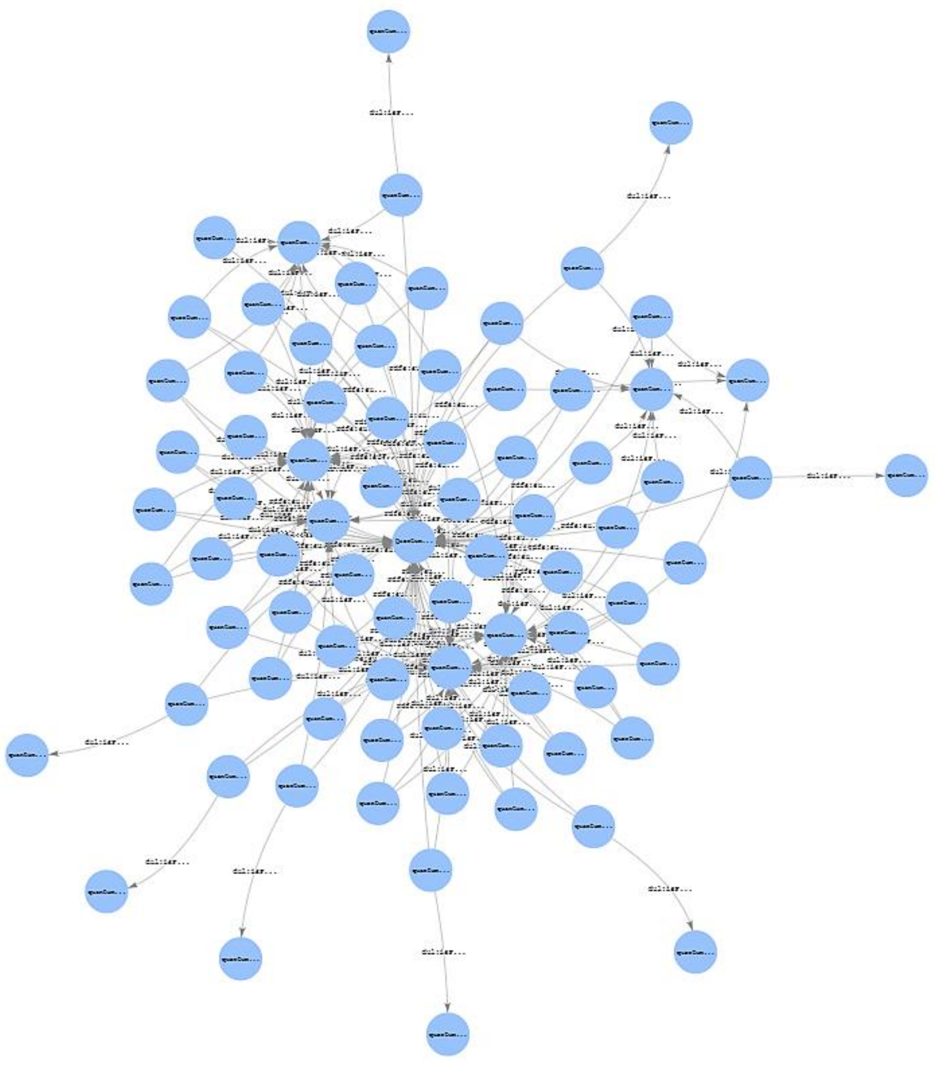


Figure 4, Quantum algorithms in QuantumShare

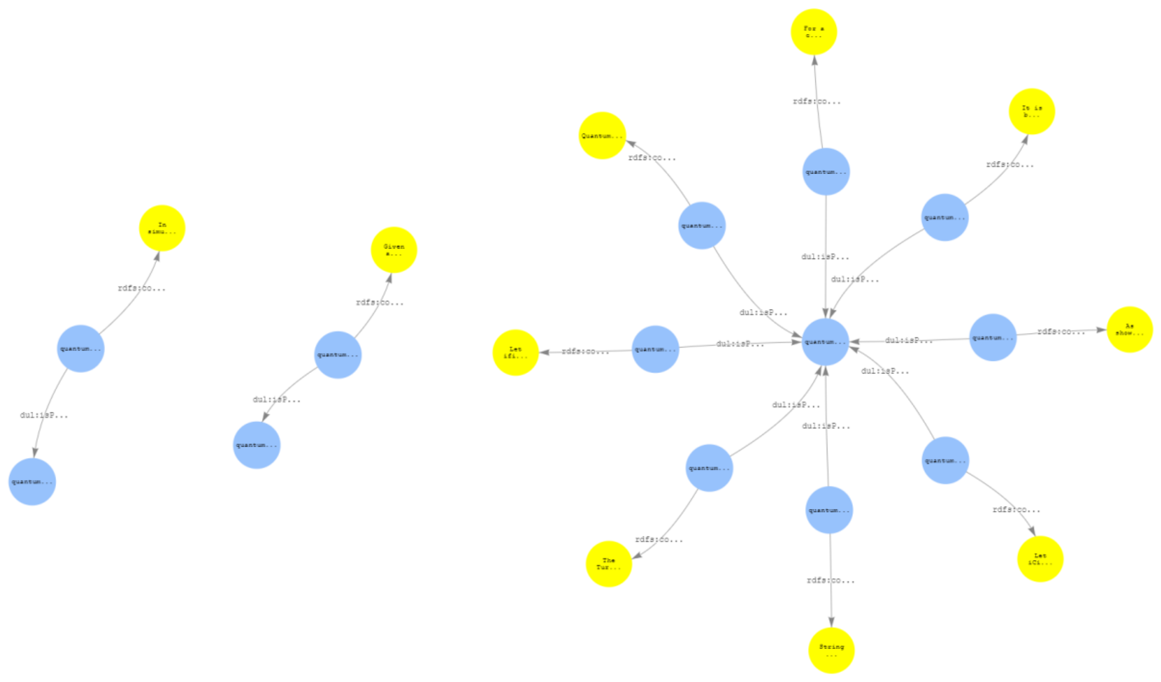


Figure 5, Quantum algorithms in QuantumShare concerning approximation and simulation

Subsequently, the developer finds out that QuantumShare contains the algorithm they are currently researching on: a quantum approximate optimization algorithm. To strengthen his literature review, he requests all scientific publications related to this algorithm, including their metadata, to facilitate look-up, as seen in Figure 34. This request corresponds with competency questions 30 and 33 and is solved by querying the corresponding SPARQL query in Table 16.

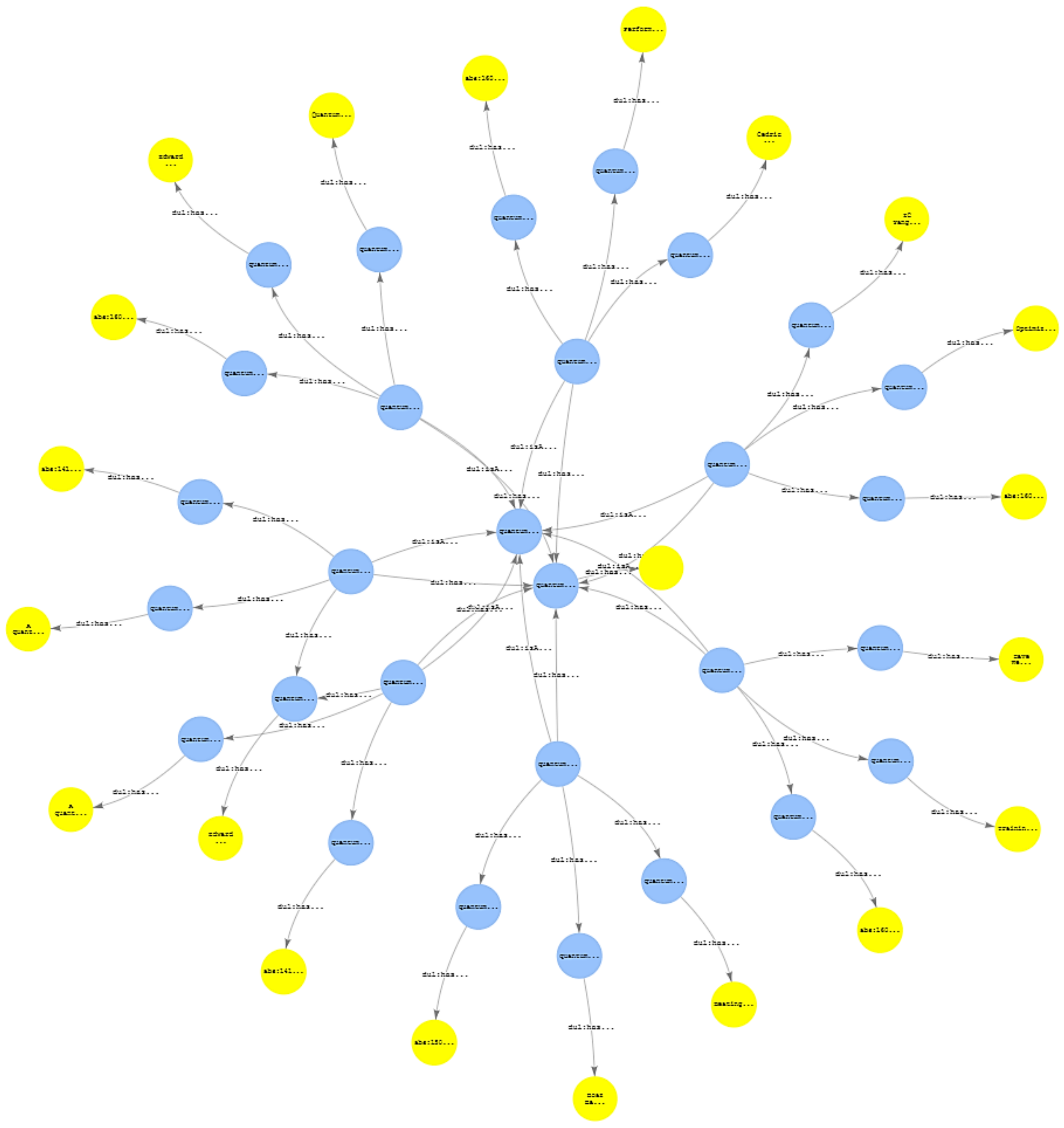


Figure 6, Quantum approximate optimization algorithm references captured in QuantumShare

An Exploration of Quantum Applications by Non-Expert Adopters

Further, in the research and development of the approximate optimization algorithm, the developer demands a practical use case to test the algorithm. Consequently, an acquaintance working in the energy and petrochemical industry is approached. However, the acquaintance is entirely unaware of quantum computing and requests an application of the approximate optimization algorithm involving a similar organization.

Table 2, Competency questions mapped to SPARQL queries for ontology testing

CQ	SPARQL query

11, 25, 28, 29

```
PREFIX quantumshare: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumShare#>
PREFIX QuantumAlgorithm: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumAlgorithm#>
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX ProblemExecution: <http://www.semanticweb.org/20173656/ontologies/2022/5/ProblemExecution#>
PREFIX org: <http://www.w3.org/ns/org#>

SELECT ?organization ?department ?site

WHERE {?execution dul:realizes quantumshare:Quantum_Approximate_Optimization .
        ?execution ProblemExecution:hasOrganization ?department .
        ?department org:unitOf ?organization .
        ?organization org:classification quantumshare:energy_provider .
         ?department org:hasSite ?site}
```

1, 3, 4, 5, 11

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX quantumshare: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumShare#>
PREFIX QuantumAlgorithm: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumAlgorithm#>
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX ProblemExecution: <http://www.semanticweb.org/20173656/ontologies/2022/5/ProblemExecution#>

CONSTRUCT {?execution dul:realizes quantumshare:Quantum_Approximate_Optimization .
        ?execution ProblemExecution:solvesProblem ?problem .
        ?problem dul:isPartOf ?context}

WHERE {?execution dul:realizes quantumshare:Quantum_Approximate_Optimization .
        ?execution ProblemExecution:solvesProblem ?problem .
        ?problem dul:isPartOf ?context}
```

26, 27

```
PREFIX quantumshare: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumShare#>
PREFIX QuantumAlgorithm: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumAlgorithm#>
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX ProblemExecution: <http://www.semanticweb.org/20173656/ontologies/2022/5/ProblemExecution#>
PREFIX mls: <http://www.w3.org/ns/mls/cp#>

CONSTRUCT {?execution mls:hasInput ?input .
         ?input dul:defines ?data .
         ?data dul:hasParameter ?dataproperties .
         ?execution mls:hasOutput ?output .
         ?output dul:defines ?performance .
         ?performance dul:hasParameter ?performanceproperties.
         ?performanceproperties dul:hasParameterDataValue ?value}

WHERE {?execution dul:realizes quantumshare:Quantum_Approximate_Optimization .
        ?execution mls:hasInput ?input .
         ?input dul:defines ?data .
         ?data dul:hasParameter ?dataproperties .
         ?execution mls:hasOutput ?output .
         ?output dul:defines ?performance .
         ?performance dul:hasParameter ?performanceproperties .
         OPTIONAL {?performanceproperties dul:hasParameterDataValue ?value}}
```

32, 33

```
PREFIX quantumshare: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumShare#>
PREFIX QuantumAlgorithm: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumAlgorithm#>
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX ProblemExecution: <http://www.semanticweb.org/20173656/ontologies/2022/5/ProblemExecution#>

SELECT ?ref ?value

WHERE {?execution dul:realizes quantumshare:Quantum_Approximate_Optimization .
        ?ref dul:isAbout ?execution .
        ?ref dul:hasParameter ?metadata.
        ?metadata dul:hasParameterDataValue ?value}
```

Firstly, they want to see the organizations involved in using quantum approximate optimization algorithms, specifically for the energy or petrochemical industry. This request requires competency questions 11, 25, 28, and 29 in the composition of a SPARQL query, which is stated in Table 17. The query finds organizations in the energy sector where a department was involved in executing the quantum approximate optimization algorithm. As a result, they discovered that ExxonMobil's research and engineering department has been experimenting with this algorithm, located at Annandale, NJ 08801, in the USA.

While becoming more interested, they wondered what problem ExxonMobil was solving by executing this optimization algorithm. They indirectly refer to competency questions 1, 3, 4, 5, and 11, satisfied by the SPARQL query denoted in Table 17. The results, which are visualized by the graph in Figure 27, show that a vehicle routing with time windows problem (VRWTW) is considered part of the vehicle routing problem class. Furthermore, VRWTW is valued as NP-hard by complexity theory, which ExxonMobil faced in the application of maritime shipping.

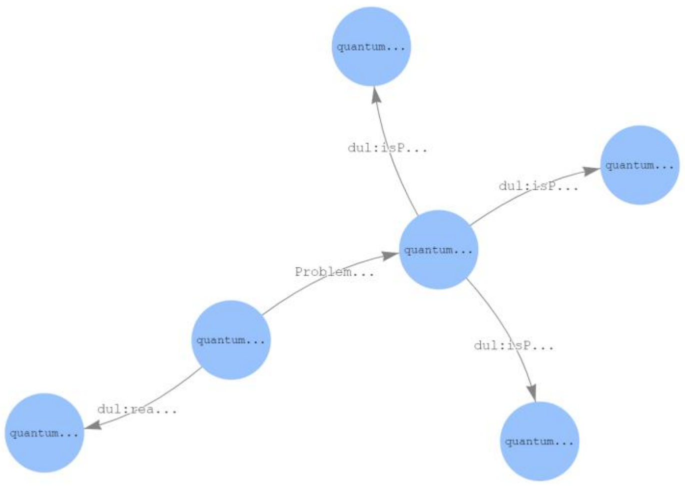


Figure 7, Problem definition related to the execution of the approximate optimization algorithm by ExxonMobil

After understanding the problem, they involve a computer scientist from the organization. The computer scientist requests additional information regarding the input and output of the execution, which refers to competency questions 26 and 27. The graph visualized in Figure 36 results from performing the related SPARQL query stated in Table 17. To the computer scientist, it becomes clear that input (left-hand side of Figure 36) consists of mathematical formulations that serve as constraints: the routes traveled, feasible movements between customers and ports, and the order of visited locations. The output is a set of evaluation metrics defined by numerical experiments (right-hand side of Figure 36): number of qubits required (value 16), number of iterations, and the probabilities of success and feasible solutions.

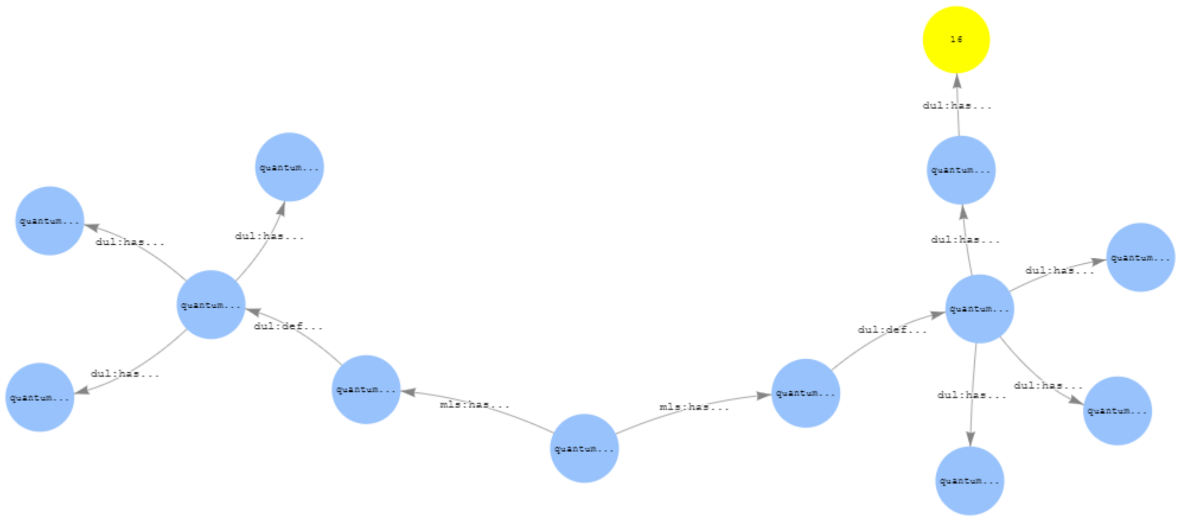


Figure 8, Input and output of the execution of the approximate optimization algorithm by ExxonMobil

The computer scientist is interested but needs additional information. Therefore, publications that reference this execution are demanded, corresponding with answering competency questions 32 and

33. As a result, two publications are retrieved by executing the SPARQL query stated in Table 17: “Formulating and Solving Routing Problems on Quantum Computers” and “Overview and Comparison of Gate Level Quantum Software Platforms”.

An Exploration of Quantum Implementations by Experts and Non-Expert Adopters

While reading the publications, the computer scientist discovered that quantum implementations go beyond his capabilities. Subsequently, they approach a consultancy firm to create an overview of a potential implementation method.

Table 3, Competency questions mapped to SPARQL queries for ontology testing

CQ	SPARQL query

24	`PREFIX quantumshare: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumShare#>`
	`PREFIX QuantumAlgorithm: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumAlgorithm#>`
	`PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>`
	`PREFIX ProblemExecution: <http://www.semanticweb.org/20173656/ontologies/2022/5/ProblemExecution#>`
	
	`SELECT ?implementation`
	
	`WHERE {?execution dul:realizes quantumshare:Quantum_Approximate_Optimization .`
	`        ?execution dul:isPartOf ?implementation}`

18, 20, 21	`PREFIX quantumshare: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumShare#>`
	`PREFIX QuantumAlgorithm: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumAlgorithm#>`
	`PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>`
	`PREFIX ProblemExecution: <http://www.semanticweb.org/20173656/ontologies/2022/5/ProblemExecution#>`
	`PREFIX cs: <http://mklab.iti.gr/pericles/ComputerSystem_ODP#>`
	`PREFIX Implementation:`
	`<http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumImplementation#>`
	`PREFIX muo: <http://elite.polito.it/ontologies/muo-vocab.owl#>`
	
	`CONSTRUCT {?implementation cs:usesHardware ?resource .`
	`            ?resource Implementation:involvesOrganization ?resourceprovider .`
	`            ?resource dul:hasParameter ?resourceproperty .`
	`            ?resourceproperty dul:hasParameterDataValue ?value .`
	`            ?resourceproperty dul:parametrizes ?quality .`
	`            ?quality muo:measuredIn ?measurementunit}`
	
	`WHERE {?execution dul:realizes quantumshare:Quantum_Approximate_Optimization .`
	`        ?execution dul:isPartOf ?implementation .`
	`        ?implementation cs:usesHardware ?resource .`
	`        ?resource rdfs:subClassOf Implementation:Quantum_Processing_Unit .`
	`        ?resource Implementation:involvesOrganization ?resourceprovider .`
	`        ?resource dul:hasParameter ?resourceproperty .`
	`        ?resourceproperty dul:hasParameterDataValue ?value .`
	`        OPTIONAL {?resourceproperty dul:parametrizes ?quality .`
	`        ?quality muo:measuredIn ?measurementunit}}`

12, 14	`PREFIX quantumshare: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumShare#>`
	`PREFIX QuantumAlgorithm: <http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumAlgorithm#>`
	`PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>`
	`PREFIX ProblemExecution: <http://www.semanticweb.org/20173656/ontologies/2022/5/ProblemExecution#>`
	`PREFIX cs: <http://mklab.iti.gr/pericles/ComputerSystem_ODP#>`
	`PREFIX Implementation:`
	`<http://www.semanticweb.org/20173656/ontologies/2022/5/QuantumImplementation#>`
	`PREFIX muo: <http://elite.polito.it/ontologies/muo-vocab.owl#>`
	
	`CONSTRUCT {?implementation cs:usesSoftware ?software .`
	`            ?externalsoftware cs:isCompatibleWith ?implementation .`
	`            ?software dul:hasParameter ?softwareproperty .`
	`            ?softwareproperty dul:hasParameterDataValue ?value .`
	`            ?softwareproperty dul:parametrizes ?quality .`
	`            ?quality muo:measuredIn ?measurementunit .`
	`            ?externalsoftware rdfs:subClassOf ?type}`
	
	`WHERE {?execution dul:realizes quantumshare:Quantum_Approximate_Optimization .`
	`        ?execution dul:isPartOf ?implementation .`
	`        ?implementation cs:usesSoftware ?software .`
	`        ?externalsoftware cs:isCompatibleWith ?implementation .`
	`        ?software dul:hasParameter ?softwareproperty .`
	`        ?softwareproperty dul:hasParameterDataValue ?value .`
	`        ?externalsoftware rdfs:subClassOf ?type .`
	`        OPTIONAL {?softwareproperty dul:parametrizes ?quality .`

```
                ?quality muo:measuredIn ?measurementunit}}
```

The dedicated consultant starts with retrieving the implementation framework used in the application of ExxonMobil, which corresponds to competency question 24. After querying the corresponding SPARQL query in Table 18, they find out that ExxonMobil's execution was implemented using Qiskit.

Afterward, they clarify the computational quantum resources compatible with Qiskit while preferring a quantum processing unit (QPU) over a local quantum simulator. A SPARQL query (Table 18) based on competency questions 18, 20, and 21 was executed to create the overview. Figure 37 visualizes the graph resulting from the query. Qiskit uses IBMQX5, a QPU offered by resource provider IBM, and this QPU has multiple properties. Firstly, it facilitates the use of 16 qubits. Secondly, it has a minimum and maximum coherence time, representing a duration measured in microseconds. These coherence times define a minimum value of 31 and a maximum of 89 microseconds. Lastly, IBM also documents gate fidelity, a fraction represented by percentages. The IBMQX5 achieves a single-qubit gate fidelity of 99.5% and a multi-qubit gate fidelity of 94.9%.

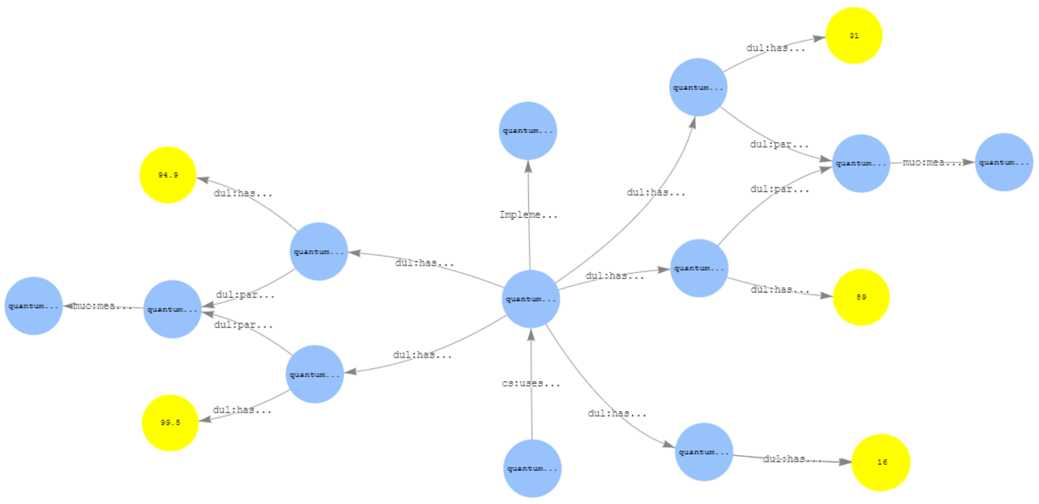


Figure 9, Quantum processing unit and its provider and resource properties, compatible with Qiskit implementation

In addition, the consultant creates an overview of the software involved in the Qiskit implementation. Satisfying competency questions 12 and 14 resulted in the graph visualization in Figure 38, facilitated by the corresponding SPARQL query in Table 18. Firstly, they identify that Qiskit software currently runs on version 0.5.4, is licensed with Apache-2.0, and can operate from Mac, Windows, and Linux systems. Furthermore, additional software can be used within the Qiskit framework. Qiskit is compatible with JavaScript, Python, and Swift (i.e., syntax programming languages), OpenQASM (i.e., assembly programming language), and its built-in Qiskit quantum programming language. Lastly, Qiskit is compatible with the Aqua software library.

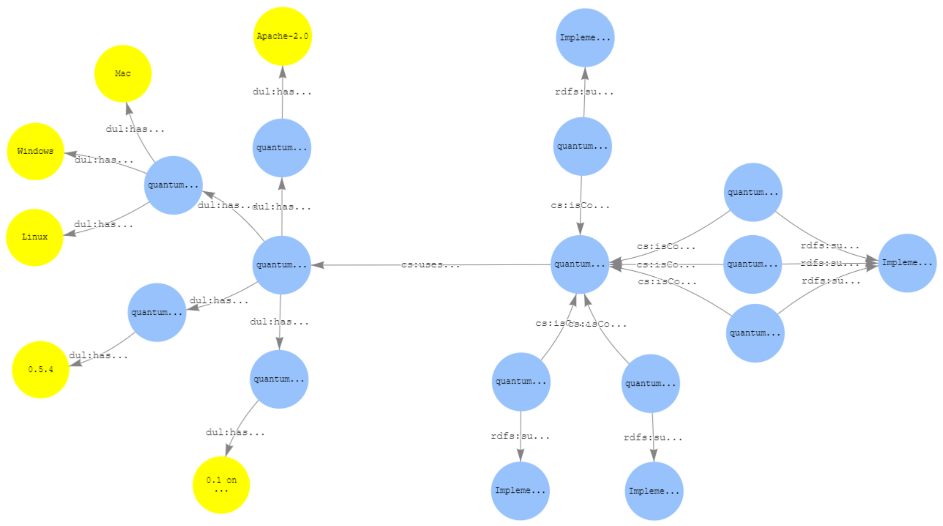


Figure 10, Software used by and compatible with Qiskit implementation