Fuzzy Inference on Quantum Annealers

Amir Pourabdollah and Colin Wilmott
School of Science and Technology
Nottingham Trent University
Nottingham, UK
amir.pourabdollah|colin.wilmott@ntu.ac.uk

Roberto Schiattarella and Giovanni Acampora
Dept. of Physics "Ettore Pancini"
University of Naples Federico II
Naples, Italy
roberto.schiattarella|giovanni.acampora@unina.it

***Abstract*—Quantum computers can potentially perform certain types of optimisation problems much more efficiently than classical computers, making them a promising tool for solving complex fuzzy logic problems. In two recent developments, based on solving Quadratic Unconstrained Binary Optimization (QUBO) problems on a type of quantum computers known as quantum annealers, we have introduced novel representations of a) fuzzy sets; b) implementations of some basic fuzzy logic operators (union, intersection, alpha-cut and maximum) and; c) the centroid defuzzification. In this paper, the previous works are further extended by presenting an implementation of Mamdani inference on the quantum annealer machines. We first present how the fuzzy rules can be formulated for such an implementation, then we present how to cascade different quantum-fuzzy operators in order to implement the quantum-fuzzy inference, and finally, a sample implementation of the inference on a real quantum computer is demonstrated. Having the main components of a rule-based fuzzy logic system implemented on quantum computers, this paper provides an integrated solution for implementing a whole fuzzy rule-based system on quantum computers.**

***Index Terms*—quantum computing, fuzzy logic, inference.**

I. Introduction

Quantum Fuzzy Logic (QFL) is a promising research field, which aims to exploit quantum computing as backend for fuzzy logic systems. The ultimate goal of this approach is the development of quantum fuzzy inference systems which can be used in environments intractable for the classical counterpart. Indeed, although classical Fuzzy Rule-Based Systems (FRBSs) have found a widespread set of applications in the field of automatic control and decision-making [1]–[4] thanks to their capability of being easily introduced in the inference process of expert knowledge by means of fuzzy rules, they suffer in data- and/or rule-intensive environments [5]. As these environments are increasingly common in current applications of fuzzy systems, there is a strong emergence of identifying innovative computational paradigms that can help to design a new generation of FRBSs capable of overcoming the above limitations. In this regard, quantum computing can be considered as a very suitable candidate. Indeed, quantum computers through quantum mechanical principles such as superposition and entanglement promise to manipulate information more efficiently than traditional electronic computers. Currently, two quantum computing paradigms are available: the first is the so-called quantum circuital model [6], and the second is the quantum annealing (or adiabatic) model [7].

Quantum annealing has developed to be a formidable challenger to quantum circuit model with promising connections of complexity theory and, especially, condensed matter physics and chemistry. However, quantum annealing and adiabatic quantum computing has a relatively short, though fruitful, history and challenges. For instance, is it possible to achieve adiabatic quantum computing with stochastic Hamiltonians, can exponential speedups be achieved using adiabatic optimization, and can we develop a theory of fault-tolerant adiabatic quantum computing? Solutions to these problems will significantly advance our understanding of quantum annealing and adiabatic quantum computing.

Both the paradigms have been exploited in the context of QFL: very recently, a Quantum Fuzzy Inference Engine (QFIE) has been developed [8] based on quantum circuits which is able to achieve an exponential speed-up in computing fuzzy rules over a classical Mamdani FRBS. On the other hand, two studies [9], [10] paved the way to the development of quantum fuzzy logic operations formulated as optimisation problems that can be solved efficiently by means of quantum annealers: in [9] we introduced a quantum representation of the fuzzy sets and a QUBO representation of the operators acting on them, such as fuzzy union, fuzzy intersection, alpha-cut and maximum, then in [10] the previous work has been extended by formulating the well-known centorid defuzzification as an Ising optimisation problem. By adding the fuzzy inference, this work proposes to combine the current and previous studies on QFL to develop the very first FRBS which can be executed on quantum annealers.

The remaining of the paper is structured as follows: section II reviews the integration of fuzzy logic and quantum computing in the literature; section III makes the paper self-contained by summarising the background required; section IV introduces the new quantum-annealers based fuzzy inference; section V shows the experiments carried out on real quantum hardware; finally, section VI concludes the paper.

II. Related Works

Quantum computing is increasingly becoming a useful backend for improving existing computational intelligence algorithms or for creating new ones [11], [12]. While evolutionary or neural quantum-based approaches are widely used in the literature [13], [14], it is only in recent years that quantum computation has started to be used in the context

of fuzzy logic. As an example in [15]–[17], quantum-inspired metaheuristics have been used to enhance the capability of the fuzzy c-means clustering . These quantum-inspired metaheuristics have been exploited also to improve the robustness of fuzzy controllers by modifying their inference performance based on quantum peculiarities [18]. On the other hand, purely quantum algorithms have recently been introduced to pave the way for the development of an entirely new generation of fuzzy systems. Two main paths have emerged: the first aims to develop quantum algorithms to directly implement fuzzy inference, while the second aims to gradually implement fuzzy logic operations in a quantum-based approach which can then be combined together to result in a quantum fuzzy system.

In the former area, [19] and [20] are two pioneering works, which are limited by the fact that the first is a purely theoretical analysis about speeding up fuzzy switching control by replacing classical operations between large matrices with quantum operations and the second requires a lookup table which models the relationship between inputs and outputs that is computed classically. Only in [8], a purely quantum fuzzy inference engine able to achieve an exponential advantage in computing fuzzy rules is proposed. This algorithm is however limited by the current quantum devices that are still severely limited by the high level of noise and small number of qubits that constitute them [21].

In the latter area, a first formulation of fuzzy t-norm and t-conorm as quantum operations is presented in [22]. Similarly, in [9], [10], it is introduced a novel representation of fuzzy sets and operators based on (QUBO) problems, which are solvable efficiently by quantum annealers. This work aims to further extend the research carried out in [9], [10] by combing for the very first time the quantum fuzzy operators defined in them and developing in this way the first quantum fuzzy systems that can be run on a quantum annealer.

III. Background

The quantum adiabatic model of a fuzzy inference proposed in this paper is based on the operational units implemented in our previous works [9], [10], including fuzzy operators (such as union, intersection, maximum and minimum) as well as a defuzzification unit. In order to make the paper self-contained, this section aims to introduce BQM problems and quantum annealers, then briefly summarise all the formalisation of the fuzzy operators implemented on quantum annealer proposed on the aforementioned works. For each operator, only the final objective function formulation is provided without details.

A. Adiabatic Quantum Computers

Quantum annealing (implemented on adiabatic quantum computers) is a universal model of computation and, in terms of computation complexity, is polynomially equivalent to its quantum circuit (i.e., standard) counterpart [23]. However, despite this, quantum annealing has received somewhat less consideration to its counterpart, with the exception of it use as a subroutine in quantum chemistry algorithms. Commercial systems offering quantum annealing have targeted high numbers of qubits with the cost that those qubits have poor coherence. There is increasing engagement with commercial quantum annealing systems and one promising line of research relates to machine learning. In particular, given that optimisation is a crucial aspect to almost all machine learning tasks, the question of whether quantum annealing is well suited to machine learning is potentially a highly significant application of quantum annealing.

Quantum annealing is a type of quantum computing that is distinct from its quantum circuit counterpart. This distinction is pronounced and most clearly demonstrated by the means in which each of these formalisms engages the computation. The quantum circuit formalism treats the computation as a discrete product of unitary gate operations. This is in contrast to quantum annealing, whereby the computation seeks to map the ground state of an initial Hamiltonian to the ground solution state of a final Hamiltonian. Quantum annealing is predicated on the quantum adiabatic theorem which explains that if a quantum system begins in a ground state, it is likely to remain in a ground state so long as the system evolves slowly. Therefore, as a consequence of this theorem, the final state is very likely the ground state of a final Hamiltonian whose ground state encodes the solution to the problem of interest.

B. BQM Problem and Quantum Annealers

Binary Quadratic Model (BQM) problems are traditionally used in computer science, with applications ranging from machine learning [24] to biology [25]. They are defined as optimisation problems formulating as follows: if $\mathcal{Q}$ is an $n \times n$ upper-triangular matrix of real weights q_{ij}, and Y is a vector of binary variables y_i, a BQM problem consists in minimizing the following objective function:

$$f(Y) = \sum_{i=1}^{n}(q_i y_i) + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(q_{ij} y_i y_j) \tag{1}$$

where q_i and q_{ij} are configurable (linear and quadratic) coefficients. For simplicity, we denoted q_{ii} as q_i. BQM encompasses both Ising and QUBO problems, with the difference that in the former case the solutions are spin solutions, i.e. $y_i \in \{-1, 1\}$ with $i \in [1, \dots n]$, whereas in the latter case the solutions are binary solutions, i.e. $y_i \in \{0, 1\}$ with $i \in [1, \dots n]$. This kind of problems can be addressed efficiently by quantum annealers. In this model, the basic information units are the so-called *quantum bits (qubits)*. While a classical bit can take 0 or 1, a qubit in its superpositioned state can take both 0 and 1 with different probabilities. Moreover, qubits can exploit the quantum mechanical phenomenon of entanglement, that happens when one qubit state depends on another one. In the quantum annealers, a setting of qubits is specialised to find the optimum solution for minimising a binary objective function [7].

The quantum annealing works out the optimum solution by means of minimising the total energy of the quantum system in an annealing process. Briefly, formulating a problem in adiabatic model is finding q_i and q_{ij}, respectively associated to

the superposition and entanglement biases, so that assignments of binary values $y_1, \ldots, y_n$ minimises the objective function, thus represents the solutions to the problem. Then during an annealing phase, the qubits are collapsed to 0 or 1 states, so that the system naturally selects its minimum possible energy. This means that the binary states of the collapsed qubits collectively provide a solution for $f(Y)$ minimisation. Similar to any quantum system, the output is probabilistic, so that the solutions made by averaging a number of runs (sampling).

C. Operational Units

As described in [9], let fuzzy sets A and B have discrete membership functions $\mu_A(x_i)$ and $\mu_B(x_i)$ over the same universe of discourse $X = \{x_1, x_2, \ldots, x_n\}$. Also let a quantum system Y exist with n qubits $y_1 ... y_n$ with a BQM objective function defined as (1), in which after the quantum annealing process, y_i values can minimise $f(Y)$. Solving the minimisation problem in quantum annealing is possible in either QUBO (binary 0/1) or Ising (-1/+1) modes. Having these assumptions, we can define the following operational units over the fuzzy sets:

1) Q.Intersection: According to [9], we choose minimum-intersection and define a new set C represented by a new qfuzzy system as:

$$C = A \cap B : \sum_{i=1}^{n} \big(\mu_C(x_i) \cdot x_i\big) \tag{2}$$

A quantum system Y with n qubits (y_i) is defined. It is proved that by defining the following BQM objective function (in QUBO mode) for system Y, its collapsed ith qubit (y_i) act as a binary switch to choose either $\mu_A(x_i)$ or $\mu_B(x_i)$ as the minimum of the two.

$$f(Y) = \sum_{i=1}^{n} \big((\mu_B(x_i) - \mu_A(x_i)) \cdot y_i\big) \tag{3}$$

$$\mu_C(x_i) = (1 - y_i) \cdot \mu_A(x_i) + y_i \cdot \mu_B(x_i) \tag{4}$$

Thus, the following coefficients can be considered for $f(Y)$:

- $q_i = \mu_B(x_i) - \mu_A(x_i)$
- $q_{ij} = 0$

2) Q.Union: This is defined similar to the Q.Intersection mechanism, as in [9]. The maximum of $\mu_A(x_i)$ and $\mu_B(x_i)$ is chosen by a binary switch y_i via minimising $f(Y)$ (in QUBO mode), in which:

- $q_i = \mu_A(x_i) - \mu_B(x_i)$
- $q_{ij} = 0$

3) Q.Max: The index of the maximum membership grade of a fuzzy set A is(are) to be marked by a binary switch y_i. According to [9], we define $f(Y)$ in a way that $y_i = 1$ if and only if $\mu_A(x_i)$ is the maximum membership grade, and 0 otherwise. It is proved that this is possible by minimising $f(Y)$ (in QUBO mode), in which:

- $q_i = -\mu_A(x_i)$
- $q_{ij} = 2$

Once the index is flagged, both the index and the maximum grade are returned.

4) Q.Min: Similar to Q.Max, finding the index of the smallest membership grade of A is possible via minimising $f(Y)$, in which:

- $q_i = 1 - \mu_A(x_i)$
- $q_{ij} = 2$

Similarly, once the index is flagged, both the index and the minimum grade are returned.

5) Q.Defuzzifier: As explained in [10], centroid defuzzification is implemented by minimising a special objective function in Ising mode. For fuzzy set A, $f(Y)$ is defined as the difference between the sum of $\mu_A(x)$ on one side of the centroid and the sum on the other side. The coefficients of $f(Y)$ are defined as:

- $q_i = 0$
- $q_{ij} = \mu_A(x_i)\mu_A(x_j) - \begin{cases} 1 + \frac{n}{4}(n-4) \text{ if } j = i+1; \\ 0 \quad \text{otherwise} \end{cases}$

In such a system, the index of a switchover between 1 and -1 in the collapsed qubits of Y (which is proved to be unique) corresponds to the index of the centroid among x_i's.

There are two other simpler operational units that are not defined in the previous works, but needed to implement the fuzzy inference. These operators do not actually need an annealing process, and can be implemented classically due to their simplicity. These are as follows:

6) Q.Singleton: This operator takes a crisp value $\hat{x}$ and makes a singleton fuzzy set $A = \{1/\hat{x}\}$ (i.e., having a single member $\hat{x}$ with $\mu_A(\hat{x}) = 1$).

7) Q.Replicate: This operator takes a crisp value $\hat{x}$ and makes a fuzzy set A, in which $\mu_A(x_i) = \hat{x}$ for all i .

IV. Implementing the Mamdani Fuzzy Inference on Quantum Annealers

This section aims to explain how each building block of the Mamdani fuzzy inference system (fuzzifier, rule strength calculator, rule output calculator, and defuzzifier) are made out of the implemented operational units discussed in sec III, and how pipelining the building blocks leads to implementing a whole Mamdani fuzzy inference.

A. Fuzzy Inference Building Blocks

In this subsection, it will be shown how each of the four building blocks of a Mamdani inference system can be made by pipelining the implemented operational units explained earlier. The importance of this step is that in practice, these blocks need computational power particularly for complex fuzzy systems with numerous inputs, rules, antecedent and consequent sets. Therefore, a quantum computing solution algorithm will be beneficial for when the ideal quantum computing platforms are available. However, the challenge is to deliver the function of each block exclusively by using the operational units stated previously.

We assume a Mamdani fuzzy inference system is to be implemented with n crisp inputs ($x_1 ... x_n$), single crisp output (y), m rules, minimum intersection, maximum union, and centroid defuzzification operators.

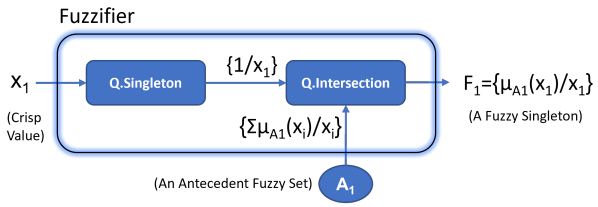

Fig. 1. The implementation of a single fuzzifier block

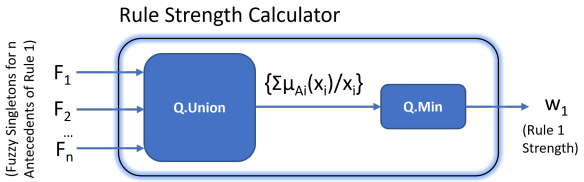

Fig. 2. The implementation of a rule firing strength calculator

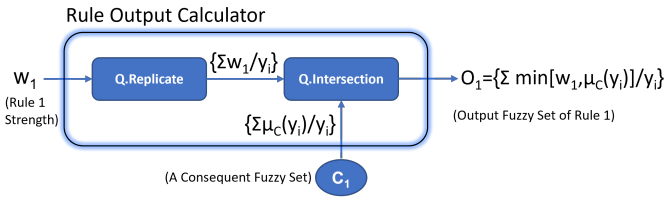

Fig. 3. The implementation of a rule output calculator block

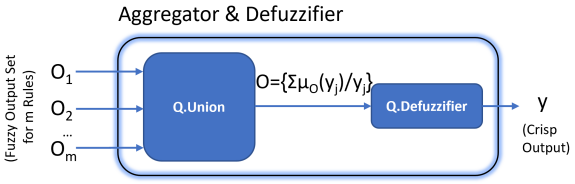

Fig. 4. Implementing the block for aggregating the individual output sets of each rule, and defuzzifying it

1) Fuzzifier: The fuzzifier block takes crisp values for each input variable, then given the antecedent fuzzy sets, it calculates the membership grade of each fuzzy set at the point of the corresponding input values. In other words, given $\hat{x}$ and a fuzzy set A, it must produce a non-normal singleton fuzzy set F with a single spike at $\hat{x}$ where $\mu_F(\hat{x}) = \mu_A(\hat{x})$. Different fuzzifier blocks are to be employed for each input and each relevant antecedent. Let us focus on a single block that converts x_1 to $F_1 = \{\Sigma\ \mu_{A_1}(x_1)/x_1\}$. The implementation of this single block is shown in Fig. 1. First, a Q.Singleton unit converts x_1 to a singleton set $\{1/x_1\}$, then a Q.Intersection unit, makes the intersection of this set with the antecedent set A_1. Since the singleton set has a single non-zero grade at x_1, the result is a singleton at x_1 with the grade $\mu_{A_1}(x_1)$.

2) Rule Strength Calculator: The job of this block is to calculate the firing strength of each rule. Each rule has n antecedents, thus the inputs of this block are n fuzzified values. In Mamdani's inference method, the firing strength of the rule is the minimum of the fuzzified values. According to our design, the inputs to the block are n fuzzy singletons coming from the fuzzifier blocks. As shown in Fig. 2, calculating the firing strength of a rule (e.g. the first rule) includes two steps: First, a union set of all the incoming fuzzified values is produced (i.e., $\{\Sigma\ \mu_{A_i}(x_i)/x_i\}$) by Q.Union unit(s). This will be a set with all the fuzzified values aggregated in a single set, in which each non-zero grade is a fuzzified value located at its corresponding crisp input. Secondly, a Q.Min unit takes the produced union and calculates the maximum grade, which is the rule's firing strength by definition (i.e., w_1 for rule 1).

3) Rule Output Calculator: Once the firing strength of all the rules are calculated, the consequent set of each rule is to be capped with the rule's strength to produce the rule's output fuzzy sets ($O_1 ... O_m$). This is equivalent to intersecting the consequent set with an intermediate "flat" fuzzy set. The strength of the rule is the membership grade of all members of such a set. This can be implemented by a Q.Replicate unit followed by a Q.Intersection unit, as shown in Fig. 3. For example in rule 1, a consequent set $C_1 = \{\Sigma\ \mu_{C_1}(y_j)/y_j\}$ is defined over its universe of discourse Y (which can be different from X). A Q.Replicate unit in this block takes the calculated rule firing strength w_1 and creates an intermediate fuzzy set $\{\Sigma\ w_1/y_j\}$ with all the grades equal to w_1. Then, this set is intersected with C_1 in order to create a capped version of C_1. Formally, the output fuzzy set of this block can be expressed as $O_1 = \{\Sigma\ min[w_1,\ \mu_{C_1}(y_j)]/y_j\}$.

4) Aggregator and Defuzzifier: After each rule's output set is created, the union of all the output sets are to be created and finally defuzzified in order to calculate the final crisp output of the inference. This can be implemented by serialising Q.Union unit(s) and a Q.Defuzzifier unit. As shown in Fig. 4, different output sets ($O_1 ... O_m$) coming from m rules are taken to the block, then their union set is produced as $O = \{\Sigma\ \mu_O(y_j)/y_j\}$ where $\mu_O(y_j) = max[\mu_{O_1}(y_j), ..., \mu_{O_m}(y_j)]$. Finally, the fuzzy output set O is to be defuzzified by a Q.Defuzzifier unit in order to calculate the final crisp output y. It is noticeable that the Q.Defuzzifier unit is an implementation of the centroid defuzzification. If more simplicity is needed, this unit can be replaced by a Q.Max that implements a MAX defuzzifier.

B. The Whole Fuzzy Inference Implementation

Once the building blocks of the inference are designed based on the quantum annealer operational units, the blocks are to be replicated and pipelined according to the number of inputs and rules, in order to create the whole inference. The proposed design of the inference system is shown in Fig. 5. As illustrated, for a system with n inputs ($A_1 ... A_n$), up to n antecedent sets can exist ($A_1 ... A_n$). Each of the m rules (e.g., rule i) can have up to n stacked fuzzifier blocks. In each rule, the multiple outputs of the fuzzifiers corresponding to the different antecedents are given to a single rule strength calculator block, in which the rule's firing strength (w_i) is calculated. Each rule also has a single consequent set C_i (which can be repeated in other rules). The calculated w_i along with the consequent set of the rule are given to a single rule

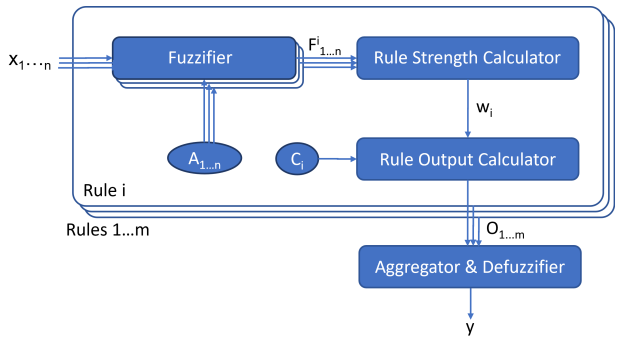

Fig. 5. Mamdani inference system made of the pipelined building blocks

output calculator block in order to produce an output set of the rule. Repeating this process for m rules produces m output sets which are to be given to a single aggregator/defuzzifier block in order to produce the final crisp output.

V. A Sample Numerical Example and Its Implementation on a Real Quantum Computer

A. A Numerical Example

Let us consider a simple fuzzy control system with two inputs, two antecedent fuzzy sets, one consequent fuzzy set and two rules. A quantum annealer with 10 qubits is assumed to be available, therefore the x-values of the discretised fuzzy sets are integers 1 to 10. Let the rules be:

- Rule 1: IF x_1 IS A_1 and x_2 IS A_2 THEN y IS C_1
- Rule 2: IF x_1 IS A_2 and x_2 IS A_1 THEN y IS C_2

and let the fuzzy sets be:

- $A_1 = \{0.0/1, 0.5/2, 1.0/3, 1.0/4, 1.0/5, 0.5/6, 0.0/7, 0.0/8, 0.0/9, 0.0/10\}$
- $A_2 = \{0.0/1, 0.0/2, 0.0/3, 0.0/4, 0.5/5, 1.0/6, 1.0/7, 1.0/8, 0.5/9, 0.0/10\}$
- $C_1 = \{1.0/1, 1.0/2, 1.0/3, 0.8/4, 0.6/5, 0.4/6, 0.2/7, 0.0/8, 0.0/9, 0.0/10\}$
- $C_2 = \{0.0/1, 0.0/2, 0.0/3, 0.2/4, 0.4/5, 0.6/6, 0.8/7, 1.0/8, 1.0/9, 1.0/10\}$

Finally, let us set the inputs as $x_1 = 2$ and $x_2 = 6$.

Each of the two rules has two fuzzifier blocks. In the first block of the first rule, with reference to Fig. 1 and 5, the steps are conducted as follows:

- The Q.Singleton unit produces a singleton set $\{1.0/2\}$
- The Q.Intersection takes the above set and A_1 to produces a fuzzy set $F_1^1 = \{0.5/2\}$

Similarly, the outputs of the other three fuzzifier blocks are: $F_1^2 = \{1.0/6\}$, $F_2^1 = \{0.0/2\}$ and $F_2^2 = \{0.5/6\}$.

The rule strength calculator block (Fig. 2) acts as follows:

- In rule 1: The Q.Union unit makes $F_1^1 \cup F_1^2 = \{0.5/2, 1.0/6\}$ then the Q.Min units calculates the rule's firing strength as $w_1 = 0.5$.
- In rule 2: The Q.Union unit makes $F_2^1 \cup F_2^2 = \{0.0/2, 0.5/6\}$ then the Q.Min unit calculates the rule's firing strength as $w_2 = 0.0$.

The rule output calculator block (Fig. 3) acts as follows:

- In rule 1: The Q.Replicate unit takes $w_1 = 0.5$ and creates a fuzzy set $\{0.5/1...10\}$, then the Q.Intersect unit produces the intersection of this set and C_1, which is $O_1 = \{0.5/1, 0.5/2, 0.5/3, 0.5/4, 0.5/5, 0.4/6, 0.2/7, 0.0/8, 0.0/9, 0.0/10\}$
- In rule 2: taking $w_2 = 0.0$, the Q.Replicate creates $\{0.5/1...10\}$, then the Q.Intersect unit produces the intersection of this set and C_2 which is $O_2 = \{0.0/1...10\}$ (i.e., rule 2 is not fired).

Finally, in the aggregator/defuzzifier block (Fig. 4) the Q.Union acts on O_1 and O_2 to create $O = 0.5/1, 0.5/2, 0.5/3, 0.5/4, 0.5/5, 0.4/6, 0.2/7, 0.0/8, 0.0/9, 0.0/10\}$, and the Q.Defuzzifier calculates the centroid of O as the final inference output y. Calculating the centroid of O yields 3.7. Given the system's resolution, the rounded centroid is $y = 4$.

B. A Sample Quantum Computer Implementation

The numerical example presented in subsection A, can be implemented on an adiabatic quantum computer with 10 qubits. The implementation is based on D-Wave System (https://docs.dwavesys.com), a cloud-based real quantum computing platform. D-Wave also provides some Python libraries for programming using web-based and desktop IDE that connect to the same platform (more details is out of the scope of this paper, and can be found in https://docs.ocean.dwavesys.com).

Each operational units explained in section II are to be implemented individually, then pipelined as shown in Fig. 5. Sample implementations of these units on a real quantum computer are already shown in [9], [10]. This paper's size does not allow to present the implementations of the whole pipelined system shown in Fig. 5. Therefore, we consider the numerical example in subsection A and show a sample implementation of a single building block, namely the final Aggregator/Defuzzifier block (Fig.4). From the example, the aggregated set is $O = \{0.5/1, 0.5/2, 0.5/3, 0.5/4, 0.5/5, 0.4/6, 0.2/7, 0.0/8, 0.0/9, 0.0/10\}$ and we implement a Q.Defuzzifier unit to calculate its centroid y.

Implementing the centroid defuzzifier on quantum annealers is detailed in [10]. Briefly, calculating the centroid is translated to an Ising-based BQM optimisation problem, of which the BQM coefficients are given in Section III-C5. According to the algorithm, when the ground states are sorted by their energy level, the centroid is found by identifying the location of the first 1/-1 switchover in the output.

Accordingly, Listing 1 shows the Python program for the Q.Defuzzifier unit. The program's output is also shown in Fig. 6, in which each row contains the states of the collapsed qubits. The rows are descendingly ranked by the system's energy levels. As explained, the defuzzified value is indicated by the first switchover point of the qubit spins in a single state (row). As highlighted in Fig. 6, the switchover is in the third row between the 3rd and the 4th qubits, i.e., the defuzzified value is a number between 3 and 4. We notice that the system's resolution is 1 and this result matches with y=3.7 that was theoretically calculated in subsection A.

	y0	y1	y2	y3	y4	y5	y6	y7	y8	y9	energy
0	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-139.92
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-139.92
2	-1	-1	-1	+1	+1	+1	+1	+1	+1	+1	-112.72
3	+1	+1	+1	-1	-1	-1	-1	-1	-1	-1	-112.72
4	+1	+1	+1	+1	-1	-1	-1	-1	-1	-1	-112.32
5	-1	-1	-1	-1	+1	+1	+1	+1	+1	+1	-112.32
6	-1	-1	+1	+1	+1	+1	+1	+1	+1	+1	-112.12
7	+1	+1	-1	-1	-1	-1	-1	-1	-1	-1	-112.12
8	-1	-1	-1	-1	-1	+1	+1	+1	+1	+1	-110.92
9	+1	+1	+1	+1	+1	-1	-1	-1	-1	-1	-110.92
10	-1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-110.52

Fig. 6. Output of Listing 1 on the real adiabatic quantum computer

Listing 1. Python program for Q.Defuzzifier unit

```
import dimod
from dwave.system import DWaveSampler,
EmbeddingComposite
mu = [0.5,0.5,0.5,0.5,0.5,0.4,0.2,0.0,0.0,0.0]
n = len(mu)
k = n*(n-4)/4+1
linear = {}
quadratic = {}
for i in range(n):
  linear.update({'y'+str(i) : 0})
for i in range(n):
  for j in range(i+1, n):
    qij=mu[i]*mu[j]
    if (j==i+1):
      qij=qij-k
    quadratic.update({('y'+str(i),'y'+str(j)):qij})
bqm = dimod.BinaryQuadraticModel(linear, quadratic,
0, 'SPIN')
sampler=EmbeddingComposite(DWaveSampler())
print (sampler.sample(bqm, num_reads=300))
```

VI. Conclusion

Considering that today's applications of fuzzy systems are increasingly involve large amounts of data or large sets of rules, there is a strong emergence of identifying innovative computational paradigms capable of efficiently managing this type of systems. This work takes another step towards modelling a whole rule-based fuzzy logic systems on quantum computers, i.e., how to orchestrate algorithms in quantum adiabatic model to achieve Mamdani's fuzzy inference.

It is important to highlight that the main goal of this paper is not to demonstrate an advantage in using quantum computers to implement a fuzzy logic system in the currently available machines. Quantum computers have the theoretical potential to solve large-scale problems, however in the current era of Noisy Intermediate-Scale Quantum (NISQ) when the real quantum computers with enough resources are not yet publicly available, classical computing methods are practically more efficient. Rather than that, the goal achieved by this research is the proof of the quantum annealers feasibility in performing a whole fuzzy inference process.

For the future, we plan to investigate the results of the proposed approach on more complex fuzzy systems - with more variables and rules. Also, some real-world application of the developed system will be explored and showcased. Moreover, this approach will be used to implement other inference methods such as zero-order TSK.

References

[1] A.-T. Nguyen, T. Taniguchi, L. Eciolaza, V. Campos, R. Palhares, and M. Sugeno, "Fuzzy control systems: Past, present and future," *IEEE Computational Intelligence Magazine*, vol. 14, no. 1, pp. 56–68, 2019.

[2] S.-M. Chen, "A new approach to handling fuzzy decision-making problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 6, pp. 1012–1016, 1988.

[3] F. J. Cabrerizo, F. Chiclana, R. Al-Hmouz, A. Morfeq, A. S. Balamash, and E. Herrera-Viedma, "Fuzzy decision making and consensus: challenges," *J. of Intell. & Fuzzy Sys.*, vol. 29, no. 3, pp. 1109–1118, 2015.

[4] Y. Liu, C. M. Eckert, and C. Earl, "A review of fuzzy ahp methods for decision-making with subjective judgements," *Expert Systems with Applications*, vol. 161, p. 113738, 2020.

[5] Y. Cui, E. Hanyu, W. Pedrycz, and Z. Li, "Designing distributed fuzzy rule-based models," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 7, pp. 2047–2053, 2020.

[6] F. Tacchino, A. Chiesa, S. Carretta, and D. Gerace, "Quantum computers as universal quantum simulators: state-of-the-art and perspectives," *Advanced Quantum Technologies*, vol. 3, no. 3, p. 1900052, 2020.

[7] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, no. 1, p. 015002, 2018.

[8] G. Acampora, R. Schiattarella, and A. Vitiello, "On the implementation of fuzzy inference engines on quantum computers," *IEEE Transactions on Fuzzy Systems*, 2022.

[9] A. Pourabdollah, G. Acampora, and R. Schiattarella, "Fuzzy logic on quantum annealers," *IEEE Transactions on Fuzzy Systems*, 2021.

[10] ——, "Implementing defuzzification operators on quantum annealers," in *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2022, pp. 1–6.

[11] A. Manju and M. J. Nigam, "Applications of quantum inspired computational intelligence: a survey," *Artificial Intelligence Review*, vol. 42, pp. 79–156, 2014.

[12] D. Ventura, "Quantum computational intelligence: answers and questions," *IEEE Intelligent Systems*, vol. 14, no. 4, pp. 14–16, 1999.

[13] G. Acampora, R. Schiattarella, and A. Vitiello, "Using quantum amplitude amplification in genetic algorithms," *Expert Systems with Applications*, vol. 209, p. 118203, 2022.

[14] Y. Kwak, W. J. Yun, S. Jung, and J. Kim, "Quantum neural networks: Concepts, applications, and challenges," in *International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2021, pp. 413–416.

[15] F. Di Martino and S. Sessa, "A novel quantum inspired genetic algorithm to initialize cluster centers in fuzzy c-means," *Expert Systems with Applications*, vol. 191, p. 116340, 2022.

[16] A.-X. Ye and Y.-X. Jin, "A fuzzy c-means clustering algorithm based on improved quantum genetic algorithm," *International Journal of Database Theory and Application*, vol. 9, no. 1, pp. 227–236, 2016.

[17] A. Baykasoğlu, İ. Gölcük, and F. B. Özsoydan, "Improving fuzzy c-means clustering via quantum-enhanced weighted superposition attraction algorithm," *Hacettepe Journal of Mathematics and Statistics*, vol. 48, no. 3, pp. 859–882, 2018.

[18] L. Litvintseva, I. Ul'yanov, S. Ul'yanov, and S. Ul'yanov, "Quantum fuzzy inference for knowledge base design in robust intelligent controllers," *Journal of Computer and Systems Sciences International*, vol. 46, no. 6, pp. 908–961, 2007.

[19] G. G. Rigatos and S. G. Tzafestas, "Parallelization of a fuzzy control algorithm using quantum computation," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 4, pp. 451–460, 2002.

[20] G. Acampora, F. Luongo, and A. Vitiello, "Quantum implementation of fuzzy systems through grover's algorithm," in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2018, pp. 1–8.

[21] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[22] L. Visintin, A. Maron, R. Reiser, and V. Kreinovich, "Aggregation operations from quantum computing," in *2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2013, pp. 1–8.

[23] D. Aharonov, W. van Dam, J. Kempe, and Landau, "Adiabatic quantum computation is equivalent to standard quantum computation," in *IEEE Symp. on Found. of Comp. Sci.* IEEE, 2004, pp. 42–51.

[24] P. Date, D. Arthur, and L. Pusey-Nazzaro, "Qubo formulations for training machine learning models," *Scientific Reports*, vol. 11, no. 1, pp. 1–10, 2021.

[25] D. M. Fox, K. M. Branson, and R. C. Walker, "mrna codon optimization with quantum computers," *PloS one*, vol. 16, no. 10, p. e0259101, 2021.